Probabilistic Databases: Introduction

EDBT-Intended Summer School

Antoine Amarilli

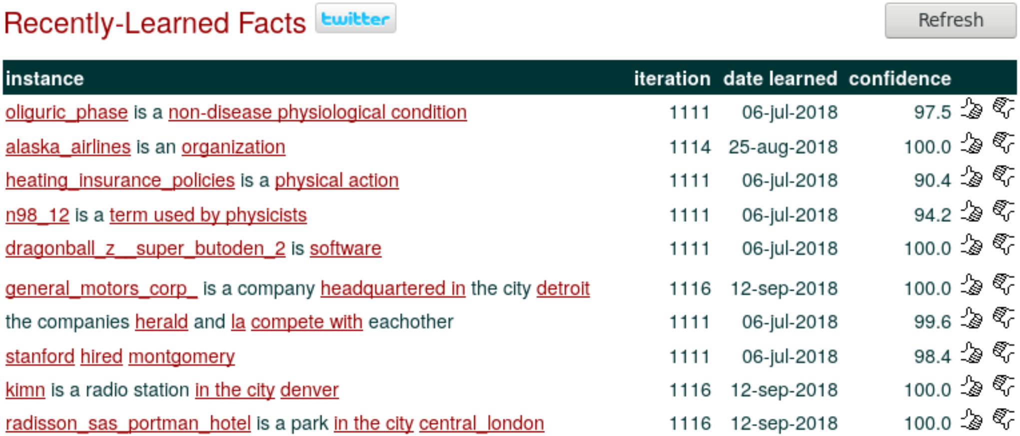
TELECOM Paris

Uncertain data: Practical motivations

Numerous sources of uncertain data:

- Measurement errors
- Data integration from contradicting sources
- Imprecise mappings between heterogeneous schemata
- Imprecise automated processes (information extraction, NLP, etc.)
- Imperfect human judgment
- Lies, opinions, rumors

Use case: Web information extraction



Recently-Learned Facts

instance	iteration	date learned	confidence
oliguric_phase is a non-disease physiological condition	1111	06-jul-2018	97.5
alaska_airlines is an organization	1114	25-aug-2018	100.0
heating_insurance_policies is a physical action	1111	06-jul-2018	90.4
n98_12 is a term used by physicists	1111	06-jul-2018	94.2
dragonball_z__super_butoden_2 is software	1111	06-jul-2018	100.0
general_motors_corp_ is a company headquartered in the city detroit	1116	12-sep-2018	100.0
the companies herald and la compete with eachother	1111	06-jul-2018	99.6
stanford hired montgomery	1111	06-jul-2018	98.4
kimn is a radio station in the city denver	1116	12-sep-2018	100.0
radisson_sas_portman_hotel is a park in the city central_london	1116	12-sep-2018	100.0

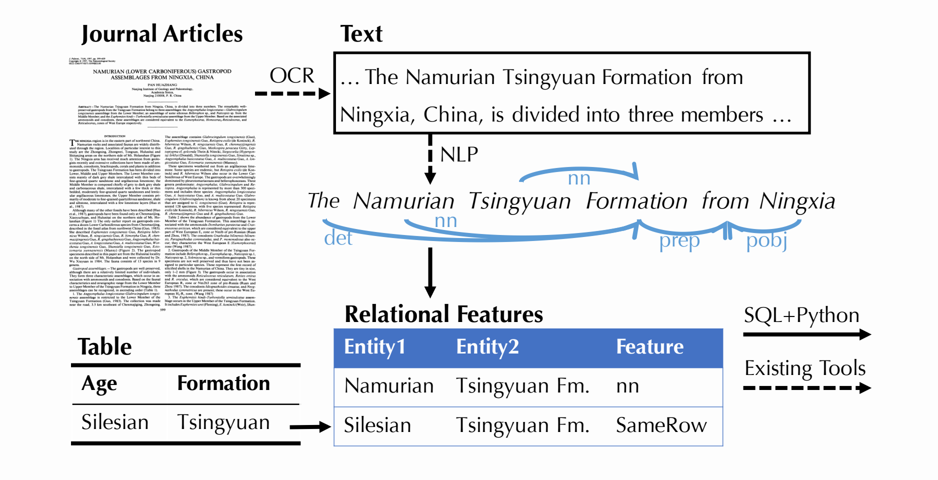
Never-ending Language Learning (NELL, CMU), `http://rtw.ml.cmu.edu/rtw/kbbrowser/`

Use case: Web information extraction

Subject	Predicate	Object	Confidence
Elvis Presley	diedOnDate	1977-08-16	97.91%
Elvis Presley	isMarriedTo	Priscilla Presley	97.29%
Elvis Presley	influences	Carlo Wolff	96.25%

YAGO, `https://www.yago-knowledge.org/`

Other use case: Information extraction from scientific articles



Journal Articles

NAMURIAN (LOWER CARBONIFEROUS) GASTROPOD ASSEMBLAGES FROM NINGXIA, CHINA

PAN HUAZHANG

OCR

Text

… The Namurian Tsingyuan Formation from Ningxia, China, is divided into three members …

NLP

The Namurian Tsingyuan Formation from Ningxia

Table

Age	Formation
Silesian	Tsingyuan

Relational Features

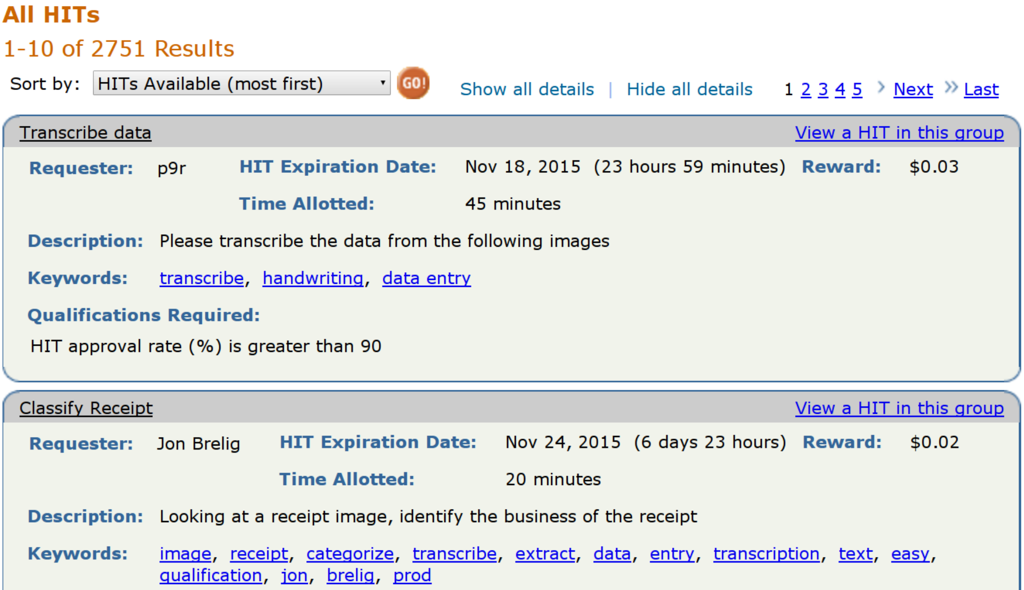
Entity1	Entity2	Feature
Namurian	Tsingyuan Fm.	nn
Silesian	Tsingyuan Fm.	SameRow

SQL+Python

Existing Tools

From GeoDeepDive / xDD

Other use case: Crowdsourcing



All HITs

1-10 of 2751 Results

Sort by: HITs Available (most first) GO!   Show all details | Hide all details   1 2 3 4 5 › Next » Last

Transcribe data   View a HIT in this group

Requester: p9r   **HIT Expiration Date:** Nov 18, 2015 (23 hours 59 minutes)   **Reward:** $0.03

Time Allotted: 45 minutes

Description: Please transcribe the data from the following images

Keywords: transcribe, handwriting, data entry

Qualifications Required:

HIT approval rate (%) is greater than 90

Classify Receipt   View a HIT in this group

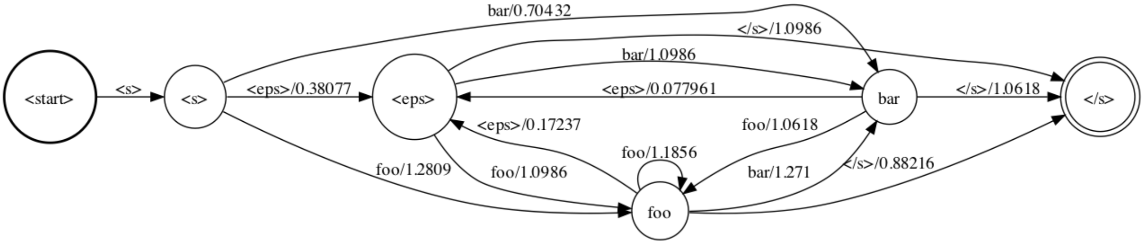
Requester: Jon Brelig   **HIT Expiration Date:** Nov 24, 2015 (6 days 23 hours)   **Reward:** $0.02

Time Allotted: 20 minutes

Description: Looking at a receipt image, identify the business of the receipt

Keywords: image, receipt, categorize, transcribe, extract, data, entry, transcription, text, easy, qualification, jon, brelig, prod

Other use case: Speech recognition and OCR



Different types of uncertainty

- The uncertainty can be **qualitative** (e.g., NULL)…
- … or **quantitative** (e.g., 95%)

Further, there are different types:

- **Unknown** value: NULL in an RDBMS
- **Alternative** between several possibilities: either A or B or C
- **Imprecision on a numeric value**: a sensor gives a value that is an approximation of the actual value
- **Confidence in a fact as a whole**: cf. information extraction
- **Structural uncertainty**: the schema of the data itself is uncertain
- **Missing data**: we know that some data is missing (open-world semantics)

What happens to this uncertainty?

Naive solution

Forget about uncertainty, or apply a threshold after each computation step

What happens to this uncertainty?

Naive solution

Forget about uncertainty, or apply a threshold after each computation step

Ideal solution

Instead of neglecting uncertainty, let’s manage it rigorously throughout the whole process of answering a query

What happens to this uncertainty?

Naive solution

Forget about uncertainty, or apply a threshold after each computation step

Ideal solution

Instead of neglecting uncertainty, let's manage it rigorously throughout the whole process of answering a query

Also: it leads to interesting theoretical questions! :)

Possible worlds semantics

Idea: use a representation system

Possible world: A regular (deterministic) relational database

Possible worlds semantics

Idea: use a representation system

Possible world: A regular (deterministic) relational database

Uncertain database: (Compact) representation of a set of possible worlds

Possible worlds semantics

Idea: use a representation system

Possible world: A regular (deterministic) relational database

Uncertain database: (Compact) representation of a set of possible worlds

Probabilistic database: (Compact) representation of a probability distribution over possible worlds,

Possible worlds semantics

Idea: use a representation system

Possible world: A regular (deterministic) relational database

Uncertain database: (Compact) representation of a set of possible worlds

Probabilistic database: (Compact) representation of a probability distribution over possible worlds, either:

- **finite:** a set of possible worlds, each with their probability
- **continuous:** more complicated

Possible worlds semantics

Idea: use a representation system

Possible world: A regular (deterministic) relational database

Uncertain database: (Compact) representation of a set of possible worlds

Probabilistic database: (Compact) representation of a probability distribution over possible worlds, either:

- **finite:** a set of possible worlds, each with their probability
- **continuous:** more complicated

date	**teacher**	
08	Diego	0.9
09	Paolo	0.8
09	Floris	0.7

Contents of this course

- Present the most common **models** of probabilistic data
  - → Focus on the **simplest one**, tuple-independent databases (TID)

Contents of this course

- Present the most common **models** of probabilistic data
  - → Focus on the **simplest one**, tuple-independent databases (TID)
- Introduce the **probabilistic query evaluation** problem (PQE):
  - → Central task: evaluating queries over probabilistic databases

Contents of this course

- Present the most common **models** of probabilistic data
  - → Focus on the **simplest one**, tuple-independent databases (TID)
- Introduce the **probabilistic query evaluation** problem (PQE):
  - → Central task: evaluating queries over probabilistic databases
- Present the **dichotomy** by Dalvi and Suciu on the complexity of PQE for UCQs

Contents of this course

- Present the most common **models** of probabilistic data
  - → Focus on the **simplest one**, tuple-independent databases (TID)
- Introduce the **probabilistic query evaluation** problem (PQE):
  - → Central task: evaluating queries over probabilistic databases
- Present the **dichotomy** by Dalvi and Suciu on the complexity of PQE for UCQs
- Present the **intensional approach** to PQE and its connections to **knowledge compilation** and **circuit classes**

Contents of this course

- Present the most common **models** of probabilistic data
  - → Focus on the **simplest one**, tuple-independent databases (TID)
- Introduce the **probabilistic query evaluation** problem (PQE):
  - → Central task: evaluating queries over probabilistic databases
- Present the **dichotomy** by Dalvi and Suciu on the complexity of PQE for UCQs
- Present the **intensional approach** to PQE and its connections to **knowledge compilation** and **circuit classes**
- Present **treewidth-based approaches** to efficient PQE

Contents of this course

- Present the most common **models** of probabilistic data
  - → Focus on the **simplest one**, tuple-independent databases (TID)
- Introduce the **probabilistic query evaluation** problem (PQE):
  - → Central task: evaluating queries over probabilistic databases
- Present the **dichotomy** by Dalvi and Suciu on the complexity of PQE for UCQs
- Present the **intensional approach** to PQE and its connections to **knowledge compilation** and **circuit classes**
- Present **treewidth-based approaches** to efficient PQE
- Give an overview of **other topics** on probabilistic databases

Probabilistic Databases: Models and PQE

EDBT-Intended Summer School

Antoine Amarilli

Relational model by example

`Guest`

`id`	`name`	`email`
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

`Reservation`

`id`	`guest`	`room`	`arrival`	`nights`
1	1	504	2022-01-01	5
2	2	107	2022-01-10	3
3	3	302	2022-01-15	6
4	2	504	2022-01-15	2
5	2	107	2022-01-30	1

Relations and databases

Formally:

- A **database schema** $\mathcal{D}$ maps each **relation name** to an **arity** (we add **attribute names** in our examples)

Relations and databases

Formally:

- A **database schema** $\mathcal{D}$ maps each **relation name** to an **arity** (we add **attribute names** in our examples)
- A **database instance** over database schema $\mathcal{D}$ maps each **relation name** R of $\mathcal{D}$ with arity k to a set of k-tuples

Relations and databases

Formally:

- A **database schema** $\mathcal{D}$ maps each **relation name** to an **arity** (we add **attribute names** in our examples)
- A **database instance** over database schema $\mathcal{D}$ maps each **relation name** R of $\mathcal{D}$ with arity k to a set of k-tuples

We can write tuples as **table rows** or as **ground facts**:

`Guest`

`id`	`name`	`email`
1	John Smith	john.smith@gmail.com
2	Alice Black	alice@black.name
3	John Smith	john.smith@ens.fr

Guest(1, John Smith, john.smith@gmail.com),
Guest(2, Alice Black, alice@black.name),
Guest(3, John Smith, john.smith@ens.fr)

Queries

- A **query** is an arbitrary **function** over database instances over a fixed schema $\mathcal{D}$
- We only study **Boolean queries**, i.e., queries returning only **true** or **false**

Queries

- A **query** is an arbitrary **function** over database instances over a fixed schema $\mathcal{D}$
- We only study **Boolean queries**, i.e., queries returning only **true** or **false**
- Example of query languages:
  - **Conjunctive queries** (CQ)
    - $\exists \bigwedge \cdots$: existentially quantified conjunctions of atoms
    - $Q : \exists x\, y\, z\, x'\, y'\; Guest(x, y, z) \wedge Guest(x', y', z)$

Queries

- A **query** is an arbitrary **function** over database instances over a fixed schema $\mathcal{D}$
- We only study **Boolean queries**, i.e., queries returning only **true** or **false**
- Example of query languages:
  - **Conjunctive queries** (CQ)
    - $\exists \bigwedge \cdots$: existentially quantified conjunctions of atoms
    - $Q : \exists x\, y\, z\, x'\, y'\ \mathit{Guest}(x, y, z) \wedge \mathit{Guest}(x', y', z)$
  - **Unions of conjunctive queries** (UCQ)
    - $\bigcup \exists \bigwedge \cdots$: unions of **CQs**

Queries

- A **query** is an arbitrary **function** over database instances over a fixed schema $\mathcal{D}$
- We only study **Boolean queries**, i.e., queries returning only **true** or **false**
- Example of query languages:
    - **Conjunctive queries** (CQ)
        - $\exists \bigwedge \cdots$: existentially quantified conjunctions of atoms
        - $Q : \exists x\, y\, z\, x'\, y'\ Guest(x, y, z) \wedge Guest(x', y', z)$
    - **Unions of conjunctive queries** (UCQ)
        - $\bigcup \exists \bigwedge \cdots$: unions of **CQs**
    - First-Order logic (FO)
    - **Monadic-Second Order** logic (MSO)

TID

Tuple-independent databases (TID)

- The **simplest** model: tuple-independent databases
- Annotate each **instance fact** with a **probability**

Tuple-independent databases (TID)

- The simplest model: tuple-independent databases
- Annotate each instance fact with a probability

date	**teacher**
08	Diego
09	Paolo
09	Floris

Tuple-independent databases (TID)

- The simplest model: tuple-independent databases
- Annotate each instance fact with a probability

date	**teacher**	
08	Diego	90%
09	Paolo	80%
09	Floris	70%

Tuple-independent databases (TID)

- The simplest model: tuple-independent databases
- Annotate each instance fact with a probability

date	**teacher**	
08	Diego	90%
09	Paolo	80%
09	Floris	70%

→ Assume independence between facts

Semantics of TID

- Each fact is **kept** or **discarded** with the indicated probability
- Probabilistic choices are **independent** across facts

Semantics of TID

- Each fact is kept or discarded with the indicated probability
- Probabilistic choices are independent across facts

date	teacher	
08	Diego	90%
09	Paolo	80%
09	Floris	70%

Semantics of TID

- Each fact is kept or discarded with the indicated probability
- Probabilistic choices are independent across facts

date	**teacher**	
08	Diego	90%
09	Paolo	80%
09	Floris	70%

date	**teacher**

Semantics of TID

- Each fact is kept or discarded with the indicated probability
- Probabilistic choices are independent across facts

date	teacher	
08	Diego	90%
09	Paolo	80%
09	Floris	70%

date	teacher
08	Diego

Semantics of TID

- Each fact is kept or discarded with the indicated probability
- Probabilistic choices are independent across facts

date	**teacher**	
08	Diego	90%
09	Paolo	80%
09	Floris	70%

date	**teacher**
08	Diego
09	Paolo

Semantics of TID

- Each fact is kept or discarded with the indicated probability
- Probabilistic choices are independent across facts

date	teacher	
08	Diego	90%
09	Paolo	80%
09	Floris	70%

date	teacher
08	Diego
09	Paolo
09	Floris

Semantics of TID

- Each fact is kept or discarded with the indicated probability
- Probabilistic choices are independent across facts

date	teacher	
08	Diego	90%
09	Paolo	80%
09	Floris	70%

date	teacher
08	Diego
09	Paolo
09	Floris

What's the probability of this possible world?

Semantics of TID

- Each fact is kept or discarded with the indicated probability
- Probabilistic choices are independent across facts

date	**teacher**	
08	Diego	90%
09	Paolo	80%
09	Floris	70%

date	**teacher**
08	Diego
09	Paolo
09	Floris

What's the probability of this possible world?

90%

Semantics of TID

- Each fact is **kept** or **discarded** with the indicated probability
- Probabilistic choices are **independent** across facts

date	**teacher**	
08	Diego	90%
09	Paolo	80%
09	Floris	70%

date	**teacher**
08	Diego
09	Paolo
09	Floris

What's the **probability** of this possible world?

$$90\% \times$$

Semantics of TID

- Each fact is kept or discarded with the indicated probability
- Probabilistic choices are independent across facts

date	teacher	
08	Diego	90%
09	Paolo	80%
09	Floris	70%

date	teacher
08	Diego
09	Paolo
09	Floris

What's the probability of this possible world?

$$90\% \times (100\% - 80\%)$$

Semantics of TID

- Each fact is kept or discarded with the indicated probability
- Probabilistic choices are independent across facts

date	**teacher**	
08	Diego	90%
09	Paolo	80%
09	Floris	70%

date	**teacher**
08	Diego
09	Paolo
09	Floris

What's the probability of this possible world?

$$90\% \times (100\% - 80\%) \times 70\%$$

Getting a probability distribution

The semantics of a TID I is a probability distribution on (non-probabilistic) databases...

→ the possible worlds are the subsets of facts of I

Getting a probability distribution

The semantics of a TID I is a probability distribution on (non-probabilistic) databases...

- → the possible worlds are the subsets of facts of I
  - → always keeping facts with probability 1

Getting a probability distribution

The semantics of a TID I is a probability distribution on (non-probabilistic) databases...

- → the possible worlds are the subsets of facts of I
  - → always keeping facts with probability 1

Formally, for a TID I, the probability of $J \subseteq I$ is:

Getting a probability distribution

The semantics of a TID I is a probability distribution on (non-probabilistic) databases...

- → the possible worlds are the subsets of facts of I
  - → always keeping facts with probability 1

Formally, for a TID I, the probability of $J \subseteq I$ is:

- product of $\Pr(F)$ for each fact F kept in J
- product of $1 - \Pr(F)$ for each fact F not kept in J

Is it a probability distribution?

Do the probabilities of the possible words always sum to 1?

Is it a probability distribution?

Do the probabilities of the possible words always sum to 1?

- Let N be the number of facts
- There are 2^N possible worlds

Is it a probability distribution?

Do the probabilities of the possible words always **sum to 1**?

- Let N be the **number of facts**
- There are 2^N **possible worlds**
- The probability of a possible world is a product which involves a factor $\mathrm{Pr}(F_i)$ or $1 - \mathrm{Pr}(F_i)$ for each fact $F_1, \ldots, F_N$

Is it a probability distribution?

Do the probabilities of the possible words always sum to 1?

- Let N be the number of facts
- There are 2^N possible worlds
- The probability of a possible world is a product which involves a factor $\Pr(F_i)$ or $1 - \Pr(F_i)$ for each fact $F_1, \ldots, F_N$

→ The sum of these probabilities is the result of expanding the expression:
$(\Pr(\mathrm{F}_1) + (1 - \Pr(\mathrm{F}_1))) \times \cdots \times (\Pr(\mathrm{F}_\mathrm{N}) + (1 - \Pr(\mathrm{F}_\mathrm{N})))$

Is it a probability distribution?

Do the probabilities of the possible words always **sum to 1**?

- Let N be the **number of facts**
- There are 2^N **possible worlds**
- The probability of a possible world is a product which involves a factor $\mathrm{Pr}(F_i)$ or $1 - \mathrm{Pr}(F_i)$ for each fact $F_1, \ldots, F_N$

→ The sum of these probabilities is the result of **expanding** the expression: $(\mathrm{Pr}(\mathrm{F}_1) + (1 - \mathrm{Pr}(\mathrm{F}_1))) \times \cdots \times (\mathrm{Pr}(\mathrm{F}_\mathrm{N}) + (1 - \mathrm{Pr}(\mathrm{F}_\mathrm{N})))$

- All factors are **equal to 1**, so the probabilities **sum to 1**

Expressiveness of TID

Can we represent all probabilistic instances with TID?

Expressiveness of TID

Can we represent **all** probabilistic instances with TID?

"The class is taught by Jane or Joe or no one but ***not both****"*

Expressiveness of TID

Can we represent **all** probabilistic instances with TID?

"The class is taught by Jane or Joe or no one but ***not both****"*

U_1

teacher
Jane

$\pi(U_1) = 80\%$

Expressiveness of TID

Can we represent **all** probabilistic instances with TID?

*"The class is taught by Jane or Joe or no one but **not both**"*

U_1

teacher
Jane
$\pi(U_1) = 80\%$

U_2

teacher
Joe
$\pi(U_2) = 10\%$

Expressiveness of TID

Can we represent **all** probabilistic instances with TID?

“The class is taught by Jane or Joe or no one but ***not both****”*

U_1

teacher
Jane

$\pi(U_1) = 80\%$

U_2

teacher
Joe

$\pi(U_2) = 10\%$

U_3

teacher

$\pi(U_3) = 10\%$

Expressiveness of TID

Can we represent **all** probabilistic instances with TID?

"The class is taught by Jane or Joe or no one but ***not both****"*

U_1

teacher
Jane

$\pi(U_1) = 80\%$

U_2

teacher
Joe

$\pi(U_2) = 10\%$

U_3

teacher

$\pi(U_3) = 10\%$

teacher
Jane
Joe

Expressiveness of TID

Can we represent **all** probabilistic instances with TID?

*"The class is taught by Jane or Joe or no one but **not both**"*

U_1

teacher
Jane

$\pi(U_1) = 80\%$

U_2

teacher
Joe

$\pi(U_2) = 10\%$

U_3

teacher

$\pi(U_3) = 10\%$

teacher	
Jane	10%
Joe	

Expressiveness of TID

Can we represent **all** probabilistic instances with TID?

“The class is taught by Jane or Joe or no one but ***not both****”*

U_1

teacher
Jane

$\pi(U_1) = 80\%$

U_2

teacher
Joe

$\pi(U_2) = 10\%$

U_3

teacher

$\pi(U_3) = 10\%$

teacher	
Jane	10%
Joe	80%

Expressiveness of TID

Can we represent **all** probabilistic instances with TID?

*"The class is taught by Jane or Joe or no one but **not both**"*

U_1

teacher
Jane

$\pi(U_1) = 80\%$

U_2

teacher
Joe

$\pi(U_2) = 10\%$

U_3

teacher

$\pi(U_3) = 10\%$

teacher	
Jane	10%
Joe	80%

→ We **cannot** forbid that both teach the class!

BID

Block-independent disjoint instances

- A more expressive framework than TID
- Call some attributes the key (underlined)

Block-independent disjoint instances

- A more expressive framework than TID
- Call some attributes the key (underlined)

U

day	**time**	**teacher**
09	AM	Paolo
09	AM	Floris
09	PM	Floris
09	PM	Paolo

Block-independent disjoint instances

- A more expressive framework than TID
- Call some attributes the key (underlined)

U

day	**time**	**teacher**
09	AM	Paolo
09	AM	Floris
09	PM	Floris
09	PM	Paolo

- The blocks are the sets of tuples with the same key

Block-independent disjoint instances

- A more expressive framework than TID
- Call some attributes the key (underlined)

U

day	**time**	**teacher**
09	AM	Paolo
09	AM	Floris
09	PM	Floris
09	PM	Paolo

- The blocks are the sets of tuples with the same key
- Each tuple has a probability

Block-independent disjoint instances

- A more expressive framework than TID
- Call some attributes the key (underlined)

U

day	**time**	**teacher**	
09	AM	Paolo	80%
09	AM	Floris	10%
09	PM	Floris	70%
09	PM	Paolo	1%

- The blocks are the sets of tuples with the same key
- Each tuple has a probability

Block-independent disjoint instances

- A more expressive framework than TID
- Call some attributes the key (underlined)

U

day	**time**	**teacher**	
09	AM	Paolo	80%
09	AM	Floris	10%
09	PM	Floris	70%
09	PM	Paolo	1%

- The blocks are the sets of tuples with the same key
- Each tuple has a probability
- Probabilities must sum up to ≤ 1 in each block

BID semantics

U

day	**time**	**teacher**	
09	AM	Paolo	80%
09	AM	Floris	10%
09	PM	Floris	70%
09	PM	Paolo	1%

BID semantics

U

day	**time**	**teacher**	
09	AM	Paolo	80%
09	AM	Floris	10%
09	PM	Floris	70%
09	PM	Paolo	1%

- For each **block**:

BID semantics

U

day	**time**	**teacher**	
09	AM	Paolo	80%
09	AM	Floris	10%
09	PM	Floris	70%
09	PM	Paolo	1%

- For each **block**:
  - Pick **one** fact according to probabilities

BID semantics

U

day	**time**	**teacher**	
09	AM	Paolo	80%
09	AM	Floris	10%
09	PM	Floris	70%
09	PM	Paolo	1%

- For each **block**:
    - Pick **one** fact according to probabilities
    - Possibly **no** fact if probabilities sum up to < 1

BID semantics

U

day	**time**	**teacher**	
09	AM	Paolo	80%
09	AM	Floris	10%
09	PM	Floris	70%
09	PM	Paolo	1%

- For each **block**:
  - Pick **one** fact according to probabilities
  - Possibly **no** fact if probabilities sum up to < 1

→ Do choices **independently** in each block

BID semantics

U

day	**time**	**teacher**	
09	AM	Paolo	80%
09	AM	Floris	10%
09	PM	Floris	70%
09	PM	Paolo	1%

U

day	**time**	**teacher**

- For each **block**:
    - Pick **one** fact according to probabilities
    - Possibly **no** fact if probabilities sum up to < 1

→ Do choices **independently** in each block

BID semantics

U

day	time	teacher	
09	AM	Paolo	80%
09	AM	Floris	10%
09	PM	Floris	70%
09	PM	Paolo	1%

U

day	time	teacher
09	AM	Paolo
09	AM	Floris

- For each **block**:
  - Pick **one** fact according to probabilities
  - Possibly **no** fact if probabilities sum up to < 1

→ Do choices **independently** in each block

BID semantics

U

day	**time**	**teacher**	
09	AM	Paolo	80%
09	AM	Floris	10%
09	PM	Floris	70%
09	PM	Paolo	1%

U

day	**time**	**teacher**
09	AM	Paolo
09	AM	Floris
09	PM	Floris
09	PM	Paolo

- For each **block**:
  - Pick **one** fact according to probabilities
  - Possibly **no** fact if probabilities sum up to < 1

→ Do choices **independently** in each block

BID captures TID

- Each TID can be expressed as a BID...

BID captures TID

- Each TID can be expressed as a BID...
  - → Take all attributes as key
  - → Each block contains a single fact

BID captures TID

- Each TID can be expressed as a BID...
  - → Take all attributes as key
  - → Each block contains a single fact

U

date	**teacher**	
09	Diego	90%
09	Paolo	80%
09	Floris	70%

Expressiveness of BID

Can we represent all probabilistic instances with BID?

Expressiveness of BID

Can we represent all probabilistic instances with BID?

"The class is taught by exactly two among Diego, Paolo, Floris."

Expressiveness of BID

Can we represent **all** probabilistic instances with BID?

"The class is taught by exactly two among Diego, Paolo, Floris."

U_1

teacher
Diego
Paolo
$\pi(U_1) = 80\%$

Expressiveness of BID

Can we represent **all** probabilistic instances with BID?

"The class is taught by exactly two among Diego, Paolo, Floris."

U_1

teacher
Diego
Paolo
$\pi(U_1) = 80\%$

U_2

teacher
Diego
Floris
$\pi(U_2) = 10\%$

Expressiveness of BID

Can we represent **all** probabilistic instances with BID?

"The class is taught by exactly two among Diego, Paolo, Floris."

U_1

teacher
Diego
Paolo
$\pi(U_1) = 80\%$

U_2

teacher
Diego
Floris
$\pi(U_2) = 10\%$

U_3

teacher
Paolo
Floris
$\pi(U_3) = 10\%$

Expressiveness of BID

Can we represent **all** probabilistic instances with BID?

"The class is taught by exactly two among Diego, Paolo, Floris."

U_1

teacher
Diego
Paolo
$\pi(U_1) = 80\%$

U_2

teacher
Diego
Floris
$\pi(U_2) = 10\%$

U_3

teacher
Paolo
Floris
$\pi(U_3) = 10\%$

→ If **teacher** is a key <u>**teacher**</u>, then TID

Expressiveness of BID

Can we represent **all** probabilistic instances with BID?

"The class is taught by exactly two among Diego, Paolo, Floris."

U_1

teacher
Diego
Paolo
$\pi(U_1) = 80\%$

U_2

teacher
Diego
Floris
$\pi(U_2) = 10\%$

U_3

teacher
Paolo
Floris
$\pi(U_3) = 10\%$

→ If **teacher** is a key $\underline{\textbf{teacher}}$, then **TID**

→ If **teacher** is not a key, then **only one fact**

Expressiveness of BID

Can we represent **all** probabilistic instances with BID?

"The class is taught by exactly two among Diego, Paolo, Floris."

U_1

teacher
Diego
Paolo

$\pi(U_1) = 80\%$

U_2

teacher
Diego
Floris

$\pi(U_2) = 10\%$

U_3

teacher
Paolo
Floris

$\pi(U_3) = 10\%$

→ If **teacher** is a key **teacher**, then **TID**

→ If **teacher** is not a key, then **only one fact**

→ We **cannot represent** this probabilistic instance as a BID

pc-tables

Boolean c-tables

- Set of Boolean variables $x_1, x_2, \ldots$
- Each fact has a condition: Variables, Boolean operators

Boolean c-tables

- Set of **Boolean variables** $x_1, x_2, \ldots$
- Each **fact** has a **condition**: Variables, Boolean operators

date	**teacher**	**room**	
04	Jane	Amphi A	$\neg x_1$
04	Joe	Amphi A	x_1
11	Jane	Amphi B	$x_2 \land \neg x_1$
11	Joe	Amphi B	$x_2 \land x_1$
11	Jane	Amphi C	$\neg x_2 \land \neg x_1$
11	Joe	Amphi C	$\neg x_2 \land x_1$

x_1	Jane is sick
x_2	Amphi B is available

pc-tables

A (Boolean) pc-table is:

- a database I where each tuple is annotated by a Boolean function on variables x_i
- a probability p_i that each variable x_i is true (assuming independence)

pc-tables

A (Boolean) **pc-table** is:

- a database I where each tuple is annotated by a **Boolean function** on variables x_i
- a **probability** p_i that each variable x_i is true (assuming independence)

Formally:

- A Boolean **valuation** ν of the variables maps each variable x_i to 0 or 1

pc-tables

A (Boolean) **pc-table** is:

- a database I where each tuple is annotated by a **Boolean function** on variables x_i
- a **probability** p_i that each variable x_i is true (assuming independence)

Formally:

- A Boolean **valuation** ν of the variables maps each variable x_i to 0 or 1
- The valuation ν defines a **possible world** I_ν of I containing the tuples whose Boolean function evaluates to **true** under ν

pc-tables

A (Boolean) **pc-table** is:

- a database I where each tuple is annotated by a **Boolean function** on variables x_i
- a **probability** p_i that each variable x_i is true (assuming independence)

Formally:

- A Boolean **valuation** ν of the variables maps each variable x_i to 0 or 1
- The valuation ν defines a **possible world** I_ν of I containing the tuples whose Boolean function evaluates to **true** under ν
- The **probability** of a valuation ν is:

pc-tables

A (Boolean) **pc-table** is:

- a database I where each tuple is annotated by a **Boolean function** on variables x_i
- a **probability** p_i that each variable x_i is true (assuming independence)

Formally:

- A Boolean **valuation** ν of the variables maps each variable x_i to 0 or 1
- The valuation ν defines a **possible world** I_ν of I containing the tuples whose Boolean function evaluates to **true** under ν
- The **probability** of a valuation ν is:
  - Product of the p_i for the x_i with $\nu(x_i) = 1$

pc-tables

A (Boolean) **pc-table** is:

- a database I where each tuple is annotated by a **Boolean function** on variables x_i
- a **probability** p_i that each variable x_i is true (assuming independence)

Formally:

- A Boolean **valuation** ν of the variables maps each variable x_i to 0 or 1
- The valuation ν defines a **possible world** I_ν of I containing the tuples whose Boolean function evaluates to **true** under ν
- The **probability** of a valuation ν is:
  - Product of the p_i for the x_i with $\nu(x_i) = 1$
  - Product of the $1 - p_i$ for the x_i with $\nu(x_i) = 0$

pc-tables

A (Boolean) **pc-table** is:

- a database I where each tuple is annotated by a **Boolean function** on variables x_i
- a **probability** p_i that each variable x_i is true (assuming independence)

Formally:

- A Boolean **valuation** ν of the variables maps each variable x_i to 0 or 1
- The valuation ν defines a **possible world** I_ν of I containing the tuples whose Boolean function evaluates to **true** under ν
- The **probability** of a valuation ν is:
  - Product of the p_i for the x_i with $\nu(x_i) = 1$
  - Product of the $1 - p_i$ for the x_i with $\nu(x_i) = 0$

$\rightarrow$ This is like TIDs

pc-tables

A (Boolean) **pc-table** is:

- a database I where each tuple is annotated by a **Boolean function** on variables x_i
- a **probability** p_i that each variable x_i is true (assuming independence)

Formally:

- A Boolean **valuation** ν of the variables maps each variable x_i to 0 or 1
- The valuation ν defines a **possible world** I_ν of I containing the tuples whose Boolean function evaluates to **true** under ν
- The **probability** of a valuation ν is:
  - Product of the p_i for the x_i with $\nu(x_i) = 1$
  - Product of the $1 - p_i$ for the x_i with $\nu(x_i) = 0$

  → This is like TIDs
- The **probability** of a possible world $J \subseteq I$ is the total probability of the valuations ν such that $I_\nu = J$

pc-table example

date	teacher	room	
04	Jane	Amphi A	$\neg x_1$
04	Joe	Amphi A	x_1
11	Jane	Amphi B	$x_2 \wedge \neg x_1$
11	Joe	Amphi B	$x_2 \wedge x_1$
11	Jane	Amphi C	$\neg x_2 \wedge \neg x_1$
11	Joe	Amphi C	$\neg x_2 \wedge x_1$

pc-table example

date	teacher	room	
04	Jane	Amphi A	$\neg x_1$
04	Joe	Amphi A	x_1
11	Jane	Amphi B	$x_2 \land \neg x_1$
11	Joe	Amphi B	$x_2 \land x_1$
11	Jane	Amphi C	$\neg x_2 \land \neg x_1$
11	Joe	Amphi C	$\neg x_2 \land x_1$

x_1	Jane is sick
x_2	Amphi B is available

pc-table example

date	**teacher**	**room**	
04	Jane	Amphi A	$\neg x_1$
04	Joe	Amphi A	x_1
11	Jane	Amphi B	$x_2 \wedge \neg x_1$
11	Joe	Amphi B	$x_2 \wedge x_1$
11	Jane	Amphi C	$\neg x_2 \wedge \neg x_1$
11	Joe	Amphi C	$\neg x_2 \wedge x_1$

x_1 Jane is sick

→ Probability 10%

x_2 Amphi B is available

→ Probability 20%

pc-table semantics example

date	**teacher**	**room**	$x_1 : 10\%, x_2 : 20\%$
04	Jane	Amphi A	$\neg x_1$
04	Joe	Amphi A	x_1
11	Jane	Amphi B	$x_2 \land \neg x_1$
11	Joe	Amphi B	$x_2 \land x_1$
11	Jane	Amphi C	$\neg x_2 \land \neg x_1$
11	Joe	Amphi C	$\neg x_2 \land x_1$

pc-table semantics example

date	**teacher**	**room**	$x_1 : 10\%, x_2 : 20\%$
04	Jane	Amphi A	$\neg x_1$
04	Joe	Amphi A	x_1
11	Jane	Amphi B	$x_2 \wedge \neg x_1$
11	Joe	Amphi B	$x_2 \wedge x_1$
11	Jane	Amphi C	$\neg x_2 \wedge \neg x_1$
11	Joe	Amphi C	$\neg x_2 \wedge x_1$

- Take ν mapping x_1 to 0 and x_2 to 1

pc-table semantics example

date	**teacher**	**room**	$x_1 : 10\%, x_2 : 20\%$
04	Jane	Amphi A	$\neg x_1$
04	Joe	Amphi A	x_1
11	Jane	Amphi B	$x_2 \land \neg x_1$
11	Joe	Amphi B	$x_2 \land x_1$
11	Jane	Amphi C	$\neg x_2 \land \neg x_1$
11	Joe	Amphi C	$\neg x_2 \land x_1$

- Take ν mapping x_1 to 0 and x_2 to 1
- Probability of ν:

pc-table semantics example

date	**teacher**	**room**	$x_1 : 10\%, x_2 : 20\%$
04	Jane	Amphi A	$\neg x_1$
04	Joe	Amphi A	x_1
11	Jane	Amphi B	$x_2 \land \neg x_1$
11	Joe	Amphi B	$x_2 \land x_1$
11	Jane	Amphi C	$\neg x_2 \land \neg x_1$
11	Joe	Amphi C	$\neg x_2 \land x_1$

- Take ν mapping x_1 to 0 and x_2 to 1
- Probability of ν: $(100\% - 10\%) \times 20\% = 18\%$

pc-table semantics example

date	**teacher**	**room**	$x_1 : 10\%, x_2 : 20\%$
04	Jane	Amphi A	$\neg x_1$
04	Joe	Amphi A	x_1
11	Jane	Amphi B	$x_2 \wedge \neg x_1$
11	Joe	Amphi B	$x_2 \wedge x_1$
11	Jane	Amphi C	$\neg x_2 \wedge \neg x_1$
11	Joe	Amphi C	$\neg x_2 \wedge x_1$

- Take ν mapping x_1 to 0 and x_2 to 1
- Probability of ν: $(100\% - 10\%) \times 20\% = 18\%$
- Evaluate the conditions

pc-table semantics example

date	**teacher**	**room**	$x_1 : 10\%, x_2 : 20\%$
04	Jane	Amphi A	$\neg x_1$
04	Joe	Amphi A	x_1
11	Jane	Amphi B	$x_2 \wedge \neg x_1$
11	Joe	Amphi B	$x_2 \wedge x_1$
11	Jane	Amphi C	$\neg x_2 \wedge \neg x_1$
11	Joe	Amphi C	$\neg x_2 \wedge x_1$

date	**teacher**	**room**
04	Jane	Amphi A
04	Joe	Amphi A
11	Jane	Amphi B
11	Joe	Amphi B
11	Jane	Amphi C
11	Joe	Amphi C

- Take ν mapping x_1 to **0** and x_2 to **1**
- Probability of ν: $(100\% - 10\%) \times 20\% = 18\%$
- Evaluate the conditions

pc-table semantics example

date	teacher	room	$x_1 : 10\%, x_2 : 20\%$
04	Jane	Amphi A	$\neg x_1$
04	Joe	Amphi A	x_1
11	Jane	Amphi B	$x_2 \land \neg x_1$
11	Joe	Amphi B	$x_2 \land x_1$
11	Jane	Amphi C	$\neg x_2 \land \neg x_1$
11	Joe	Amphi C	$\neg x_2 \land x_1$

date	teacher	room
04	Jane	Amphi A
04	Joe	Amphi A
11	Jane	Amphi B
11	Joe	Amphi B
11	Jane	Amphi C
11	Joe	Amphi C

- Take ν mapping x_1 to 0 and x_2 to 1
- Probability of ν: $(100\% - 10\%) \times 20\% = 18\%$
- Evaluate the conditions

→ Probability of possible world: sum over the valuations

pc-table semantics example

date	**teacher**	**room**	x_1 : 10%, x_2 : 20%
04	Jane	Amphi A	$\neg x_1$
04	Joe	Amphi A	x_1
11	Jane	Amphi B	$x_2 \wedge \neg x_1$
11	Joe	Amphi B	$x_2 \wedge x_1$
11	Jane	Amphi C	$\neg x_2 \wedge \neg x_1$
11	Joe	Amphi C	$\neg x_2 \wedge x_1$

date	**teacher**	**room**
04	Jane	Amphi A
04	Joe	Amphi A
11	Jane	Amphi B
11	Joe	Amphi B
11	Jane	Amphi C
11	Joe	Amphi C

- Take ν mapping x_1 to **0** and x_2 to **1**
- **Probability** of ν: $(100\% - 10\%) \times 20\% = 18\%$
- Evaluate the **conditions**

→ Probability of possible world: **sum** over the valuations

  → Here: **only** this valuation, 18%

Expressiveness of pc-tables

- pc-tables capture TIDs:
  - → Simply give each fact its own probability value

Expressiveness of pc-tables

- pc-tables capture **TIDs**:
  - → Simply give each fact its own **probability value**

- pc-tables capture **BIDs**:
  - → Make a **decision tree** for every block

Expressiveness of pc-tables

- pc-tables capture **TIDs**:
  - → Simply give each fact its own **probability value**

- pc-tables capture **BIDs**:
  - → Make a **decision tree** for every block

- In fact pc-tables can express **arbitrary probability distributions**

Expressiveness of pc-tables

- pc-tables capture **TIDs**:
  → Simply give each fact its own **probability value**
- pc-tables capture **BIDs**:
  → Make a **decision tree** for every block
- In fact pc-tables can express **arbitrary probability distributions**
- Further, they are a **strong representation system**: the union, product, etc., of two pc-tables, can be easily represented as a pc-table

Expressiveness of pc-tables

- pc-tables capture **TIDs**:
  - → Simply give each fact its own **probability value**
- pc-tables capture **BIDs**:
  - → Make a **decision tree** for every block
- In fact pc-tables can express **arbitrary probability distributions**
- Further, they are a **strong representation system**: the union, product, etc., of two pc-tables, can be easily represented as a pc-table

Yet, in the rest of the talk, we focus on **TIDs** → easier to characterize tractable queries

PQE

Query evaluation on probabilistic databases (PQE)

How can we evaluate a query Q over a probabilistic database D?

Query evaluation on probabilistic databases (PQE)

How can we evaluate a query *Q* over a probabilistic database *D*?

- Probability that *Q* holds over *D*:

$$\Pr(D \models Q) = \sum_{\substack{D' \subseteq D \\ D' \models Q}} \Pr(D')$$

- **Intuitively:** the probability that Q holds is the probability of drawing a possible world $D' \subseteq D$ which satisfies Q

Query evaluation on probabilistic databases (PQE)

How can we evaluate a query Q over a probabilistic database D?

- Probability that Q holds over D:

$$\Pr(D \models Q) = \sum_{\substack{D' \subseteq D \\ D' \models Q}} \Pr(D')$$

- **Intuitively:** the probability that Q holds is the probability of drawing a possible world $D' \subseteq D$ which satisfies Q

Probabilistic query evaluation (PQE) problem for a query Q over TIDs: given a TID, compute the probability that Q holds

Example of PQE on TID

name	position	city	classification	**prob**
John	Director	New York	unclassified	0.5
Paul	Janitor	New York	restricted	0.7
Dave	Analyst	Paris	confidential	0.3
Ellen	Field agent	Berlin	secret	0.2
Magdalen	Double agent	Paris	top secret	1.0
Nancy	HR director	Paris	restricted	0.8
Susan	Analyst	Berlin	secret	0.2

What is the probability to have a tuple with value **New York**?

Example of PQE on TID

name	position	city	classification	prob
John	Director	New York	unclassified	0.5
Paul	Janitor	New York	restricted	0.7
Dave	Analyst	Paris	confidential	0.3
Ellen	Field agent	Berlin	secret	0.2
Magdalen	Double agent	Paris	top secret	1.0
Nancy	HR director	Paris	restricted	0.8
Susan	Analyst	Berlin	secret	0.2

What is the probability to have a tuple with value New York?

- It is one minus the probability of not having such a tuple

Example of PQE on TID

name	position	city	classification	prob
John	Director	New York	unclassified	0.5
Paul	Janitor	New York	restricted	0.7
Dave	Analyst	Paris	confidential	0.3
Ellen	Field agent	Berlin	secret	0.2
Magdalen	Double agent	Paris	top secret	1.0
Nancy	HR director	Paris	restricted	0.8
Susan	Analyst	Berlin	secret	0.2

What is the probability to have a tuple with value **New York**?

- It is **one minus** the probability of not having such a tuple
- Not having such a tuple is the **independent AND** of not having each tuple

Example of PQE on TID

name	position	city	classification	prob
John	Director	New York	unclassified	0.5
Paul	Janitor	New York	restricted	0.7
Dave	Analyst	Paris	confidential	0.3
Ellen	Field agent	Berlin	secret	0.2
Magdalen	Double agent	Paris	top secret	1.0
Nancy	HR director	Paris	restricted	0.8
Susan	Analyst	Berlin	secret	0.2

What is the probability to have a tuple with value **New York**?

- It is **one minus** the probability of not having such a tuple
- Not having such a tuple is the **independent AND** of not having each tuple
- So the result is $1 - (1 - 0.5) \times (1 - 0.7) = 0.85$

Complexity of PQE

Formal question:

- We fix a Boolean query, e.g., $\exists xy\, R(x), S(x,y), T(y)$

Complexity of PQE

Formal question:

- We **fix** a Boolean query, e.g., $\exists xy\ R(x), S(x,y), T(y)$
- We are given a **tuple-independent database** D, i.e., a relational database where facts are independent and have probabilities

Complexity of PQE

Formal question:

- We **fix** a Boolean query, e.g., $\exists xy\ R(x), S(x, y), T(y)$
- We are given a **tuple-independent database** D, i.e., a relational database where facts are independent and have probabilities
- Can we **compute** the total probability of the possible worlds of D that satisfy Q?

Complexity of PQE

Formal question:

- We fix a Boolean query, e.g., $\exists xy\ R(x), S(x,y), T(y)$
- We are given a tuple-independent database D, i.e., a relational database where facts are independent and have probabilities
- Can we compute the total probability of the possible worlds of D that satisfy Q?
- Note that we study data complexity, i.e., Q is fixed and the input is D

Naive probabilistic query evaluation

- Consider all possible worlds of the input

Naive probabilistic query evaluation

- Consider all possible worlds of the input
- Run the query over each possible world

Naive probabilistic query evaluation

- Consider all possible worlds of the input
- Run the query over each possible world
- Sum the probabilities of all worlds that satisfy the query

Naive probabilistic query evaluation example

TID D

in	out	
A	B	0.8
B	C	0.2

Query Q

$$R(x, y) \land R(y, z)$$

Naive probabilistic query evaluation example

TID D

in	out	
A	B	0.8
B	C	0.2

Query Q

$R(x, y) \wedge R(y, z)$

Possible worlds and probabilities:

in	out
A	B
B	C

0.8×0.2

in	out
A	B
B	C

$(1 - 0.8) \times 0.2$

in	out
A	B
B	C

$0.8 \times (1 - 0.2)$

in	out
A	B
B	C

$(1 - 0.8) \times (1 - 0.2)$

Naive probabilistic query evaluation example

TID *D*

in	out	
A	B	0.8
B	C	0.2

Query *Q*

$R(x, y) \wedge R(y, z)$

Possible worlds and probabilities:

in	out
A	B
B	C

0.8×0.2

in	out
A	B
B	C

$(1 - 0.8) \times 0.2$

in	out
A	B
B	C

$0.8 \times (1 - 0.2)$

in	out
A	B
B	C

$(1 - 0.8) \times (1 - 0.2)$

Total probability that Q holds: $0.8 \times 0.2 = 0.16$.

Naive evaluation advantages and drawbacks

- Naive evaluation is always possible

Naive evaluation advantages and drawbacks

- Naive evaluation is **always possible**
- However, it takes **exponential time** in general
  - → Even if the query output has **few possible worlds**!
  - → Feasible if the **input** has few possible worlds (few tuples)

Naive evaluation advantages and drawbacks

- Naive evaluation is **always possible**
- However, it takes **exponential time** in general
  - → Even if the query output has **few possible worlds**!
  - → Feasible if the **input** has few possible worlds (few tuples)
- In fact, naive evaluation is in **#P**
  - → Can be expressed (up to normalization) as the **number of accepting paths** of a **nondeterministic PTIME Turing machine**
  - → To see why: **guess** a possible world (with the right probabilities) and **check** the query

Naive evaluation advantages and drawbacks

- Naive evaluation is **always possible**
- However, it takes **exponential time** in general
  - → Even if the query output has **few possible worlds**!
  - → Feasible if the **input** has few possible worlds (few tuples)
- In fact, naive evaluation is in **#P**
  - → Can be expressed (up to normalization) as the **number of accepting paths** of a **nondeterministic PTIME Turing machine**
  - → To see why: **guess** a possible world (with the right probabilities) and **check** the query
- Probabilistic query evaluation is **computationally intractable** so it is unlikely that we can beat naive evaluation **in general**

Naive evaluation advantages and drawbacks

- Naive evaluation is **always possible**
- However, it takes **exponential time** in general
  - → Even if the query output has **few possible worlds**!
  - → Feasible if the **input** has few possible worlds (few tuples)
- In fact, naive evaluation is in **#P**
  - → Can be expressed (up to normalization) as the **number of accepting paths** of a **nondeterministic PTIME Turing machine**
  - → To see why: **guess** a possible world (with the right probabilities) and **check** the query
- Probabilistic query evaluation is **computationally intractable** so it is unlikely that we can beat naive evaluation **in general**
  - → But **some queries** admit an efficient algorithm!

Some examples of PQE

- What is the probability of the query: $\exists x\ R(x)$?

Some examples of PQE

- What is the probability of the query: $\exists x\ R(x)$?
  - It asks: "do we have an R-fact?"

Some examples of PQE

- What is the probability of the query: $\exists x\ R(x)$?
  - It asks: "do we have an R-fact?"

$\rightarrow$ It is:

Some examples of PQE

- What is the probability of the query: $\exists x\, R(x)$?
  - It asks: “do we have an R-fact?”

$\rightarrow$ It is: $\mathbf{1} -$

Some examples of PQE

- What is the probability of the query: $\exists x\ R(x)$?
  - It asks: “do we have an R-fact?”

$\rightarrow$ It is: $\mathbf{1} - \prod_{R(a)}$

Some examples of PQE

- What is the probability of the query: $\exists x\, R(x)$?
  - It asks: "**do we have an *R*-fact?**"

→ It is: $1 - \prod_{R(a)} (1 - \mathrm{Pr}(R(a)))$

Some examples of PQE

- What is the probability of the query: $\exists x\ R(x)$?
  - It asks: "do we have an R-fact?"

  → It is: $1 - \prod_{R(a)} (1 - \Pr(R(a)))$

- What is the probability of the query: $\exists xy\ R(x), S(x, y)$?

Some examples of PQE

- What is the probability of the query: $\exists x\ R(x)$?
    - It asks: "do we have an R-fact?"

  → It is: $1 - \prod_{R(a)} (1 - \Pr(R(a)))$

- What is the probability of the query: $\exists xy\ R(x), S(x, y)$?
    - It asks: "is there an R-fact which also has an S-fact?"

Some examples of PQE

- What is the probability of the query: $\exists x\ R(x)$?
  - It asks: "do we have an R-fact?"
  - → It is: $1 - \prod_{R(a)} (1 - \Pr(R(a)))$

- What is the probability of the query: $\exists xy\ R(x), S(x, y)$?
  - It asks: "is there an R-fact which also has an S-fact?"
  - Idea: case disjunction based on the value of x

Some examples of PQE

- What is the probability of the query: $\exists x\ R(x)$?
  - It asks: "do we have an R-fact?"
  - → It is: $1 - \prod_{R(a)} (1 - \Pr(R(a)))$
- What is the probability of the query: $\exists xy\ R(x), S(x, y)$?
  - It asks: "is there an R-fact which also has an S-fact?"
  - Idea: case disjunction based on the value of x
  - We get:

Some examples of PQE

- What is the probability of the query: $\exists x \; R(x)$?
  - It asks: "**do we have an *R*-fact?**"

  → It is: $1 - \prod_{R(a)} (1 - \Pr(R(a)))$

- What is the probability of the query: $\exists xy \; R(x), S(x,y)$?
  - It asks: "**is there an *R*-fact which also has an *S*-fact?**"
  - Idea: **case disjunction** based on the value of x
  - We get: $1 -$

Some examples of PQE

- What is the probability of the query: $\exists x\ R(x)$?
  - It asks: "**do we have an *R*-fact?**"
  - → It is: $1 - \prod_{R(a)} (1 - \Pr(R(a)))$

- What is the probability of the query: $\exists xy\ R(x), S(x,y)$?
  - It asks: "**is there an *R*-fact which also has an *S*-fact?**"
  - Idea: **case disjunction** based on the value of x
  - We get: $1 - \prod_a$

Some examples of PQE

- What is the probability of the query: $\exists x\ R(x)$?
  - It asks: "do we have an *R*-fact?"
  - → It is: $1 - \prod_{R(a)} (1 - \Pr(R(a)))$

- What is the probability of the query: $\exists xy\ R(x), S(x, y)$?
  - It asks: "is there an *R*-fact which also has an *S*-fact?"
  - Idea: case disjunction based on the value of x
  - We get: $1 - \prod_{a} \big(1 -$

Some examples of PQE

- What is the probability of the query: $\exists x\ R(x)$?
  - It asks: "**do we have an R-fact?**"
  - $\rightarrow$ It is: $1 - \prod_{R(a)} (1 - \Pr(R(a)))$

- What is the probability of the query: $\exists xy\ R(x), S(x,y)$?
  - It asks: "**is there an R-fact which also has an S-fact?**"
  - Idea: **case disjunction** based on the value of x
  - We get: $1 - \prod_{a} \big(1 - \Pr(R(a)) \times$

Some examples of PQE

- What is the probability of the query: $\exists x\ R(x)$?
  - It asks: "do we have an *R*-fact?"
  - → It is: $1 - \prod_{R(a)} (1 - \mathrm{Pr}(R(a)))$

- What is the probability of the query: $\exists xy\ R(x), S(x, y)$?
  - It asks: "is there an *R*-fact which also has an *S*-fact?"
  - Idea: case disjunction based on the value of x
  - We get: $1 - \prod_a \big(1 - \mathrm{Pr}(R(a)) \times \big(1 -$

Some examples of PQE

- What is the probability of the query: $\exists x\ R(x)$?
  - It asks: "do we have an *R*-fact?"
  - → It is: $1 - \prod_{R(a)} (1 - \mathrm{Pr}(R(a)))$

- What is the probability of the query: $\exists xy\ R(x), S(x, y)$?
  - It asks: "is there an *R*-fact which also has an *S*-fact?"
  - Idea: case disjunction based on the value of x
  - We get: $1 - \prod_a \big(1 - \mathrm{Pr}(R(a)) \times \big(1 - \prod_b$

Some examples of PQE

- What is the probability of the query: $\exists x\ R(x)$?
  - It asks: "do we have an R-fact?"
  - → It is: $1 - \prod_{R(a)} (1 - \Pr(R(a)))$

- What is the probability of the query: $\exists xy\ R(x), S(x, y)$?
  - It asks: "is there an R-fact which also has an S-fact?"
  - Idea: case disjunction based on the value of x
  - We get: $1 - \prod_a \left(1 - \Pr(R(a)) \times \left(1 - \prod_b (1 -\right.\right.$

Some examples of PQE

- What is the probability of the query: $\exists x\ R(x)$?
  - It asks: "do we have an R-fact?"

  → It is: $1 - \prod_{R(a)} (1 - \Pr(R(a)))$

- What is the probability of the query: $\exists xy\ R(x), S(x, y)$?
  - It asks: "is there an R-fact which also has an S-fact?"
  - Idea: case disjunction based on the value of x
  - We get: $1 - \prod_a \left(1 - \Pr(R(a)) \times \left(1 - \prod_b (1 - \Pr(S(a, b)))\right)\right)$

Some examples of PQE

- What is the probability of the query: $\exists x\ R(x)$?
  - It asks: "**do we have an R-fact?**"
  - $\rightarrow$ It is: $1 - \prod_{R(a)} (1 - \Pr(R(a)))$

- What is the probability of the query: $\exists xy\ R(x), S(x, y)$?
  - It asks: "**is there an R-fact which also has an S-fact?**"
  - Idea: **case disjunction** based on the value of x
  - We get: $1 - \prod_a \left(1 - \Pr(R(a)) \times \left(1 - \prod_b (1 - \Pr(S(a, b)))\right)\right)$
  - Make sure you understand **why** everything is independent in this case!

Some examples of PQE

- What is the probability of the query: $\exists x\ R(x)$?
  - It asks: "do we have an R-fact?"
  - → It is: $1 - \prod_{R(a)} (1 - \Pr(R(a)))$

- What is the probability of the query: $\exists xy\ R(x), S(x, y)$?
  - It asks: "is there an R-fact which also has an S-fact?"
  - Idea: case disjunction based on the value of x
  - We get: $1 - \prod_a \left(1 - \Pr(R(a)) \times \left(1 - \prod_b (1 - \Pr(S(a, b)))\right)\right)$
  - Make sure you understand why everything is independent in this case!

- What is the probability of the query: $\exists xy\ R(x), S(x, y), T(y)$?

Some examples of PQE

- What is the probability of the query: $\exists x\ R(x)$?
  - It asks: "**do we have an R-fact?**"

  → It is: $1 - \prod_{R(a)} (1 - \Pr(R(a)))$

- What is the probability of the query: $\exists xy\ R(x), S(x,y)$?
  - It asks: "**is there an R-fact which also has an S-fact?**"
  - Idea: **case disjunction** based on the value of x
  - We get: $1 - \prod_a \left(1 - \Pr(R(a)) \times \left(1 - \prod_b (1 - \Pr(S(a,b)))\right)\right)$
  - Make sure you understand **why** everything is independent in this case!

- What is the probability of the query: $\exists xy\ R(x), S(x,y), T(y)$?
  - This one is **#P-hard**!

Some examples of PQE

- What is the probability of the query: $\exists x\ R(x)$?
  - It asks: "**do we have an *R*-fact?**"
  - → It is: $1 - \prod_{R(a)} (1 - \Pr(R(a)))$

- What is the probability of the query: $\exists xy\ R(x), S(x, y)$?
  - It asks: "**is there an *R*-fact which also has an *S*-fact?**"
  - Idea: **case disjunction** based on the value of x
  - We get: $1 - \prod_a \left(1 - \Pr(R(a)) \times \left(1 - \prod_b (1 - \Pr(S(a, b)))\right)\right)$
  - Make sure you understand **why** everything is independent in this case!

- What is the probability of the query: $\exists xy\ R(x), S(x, y), T(y)$?
  - This one is **#P-hard**!

Research question: can we **characterize the easy cases** and **prove hardness otherwise**?

Probabilistic Databases: The Dichotomy of PQE

EDBT-Intended Summer School

Antoine Amarilli

Research goal: Understanding the complexity of PQE

What is the complexity of $\mathrm{PQE}(Q)$ depending on the query Q?

Research goal: Understanding the complexity of PQE

What is the complexity of $\mathrm{PQE}(Q)$ depending on the query Q?

$\rightarrow$ Recall that we study **data complexity**, i.e., Q is **fixed** and the input is the **data**

Research goal: Understanding the complexity of PQE

What is the complexity of $\mathrm{PQE}(Q)$ depending on the query Q?

→ Recall that we study **data complexity**, i.e., Q is **fixed** and the input is the **data**

For example:

- For $Q : R(x)$ the problem is **easy** (PTIME)
- For $Q : R(x), S(x, y), T(y)$ the problem is **hard** (#P-hard)

Research goal: Understanding the complexity of PQE

What is the complexity of $\mathrm{PQE}(Q)$ depending on the query Q?

→ Recall that we study **data complexity**, i.e., Q is **fixed** and the input is the **data**

For example:

- For $Q : R(x)$ the problem is **easy** (PTIME)
- For $Q : R(x), S(x, y), T(y)$ the problem is **hard** (#P-hard)

We will present the **dichotomy** of [Dalvi and Suciu, 2007, Dalvi and Suciu, 2012]:

Research goal: Understanding the complexity of PQE

What is the complexity of $\mathrm{PQE}(Q)$ depending on the query Q?

$\rightarrow$ Recall that we study **data complexity**, i.e., Q is **fixed** and the input is the **data**

For example:

- For $Q : R(x)$ the problem is **easy** (PTIME)
- For $Q : R(x), S(x, y), T(y)$ the problem is **hard** (#P-hard)

We will present the **dichotomy** of [Dalvi and Suciu, 2007, Dalvi and Suciu, 2012]:

- Small dichotomy: **conjunctive queries** that are **self-join-free** and **arity-two**

Research goal: Understanding the complexity of PQE

What is the complexity of $\mathrm{PQE}(Q)$ depending on the query Q?

$\rightarrow$ Recall that we study **data complexity**, i.e., Q is **fixed** and the input is the **data**

For example:

- For $Q : R(x)$ the problem is **easy** (PTIME)
- For $Q : R(x), S(x, y), T(y)$ the problem is **hard** (#P-hard)

We will present the **dichotomy** of [Dalvi and Suciu, 2007, Dalvi and Suciu, 2012]:

- Small dichotomy: **conjunctive queries** that are **self-join-free** and **arity-two**
- Large dichotomy: arbitrary **unions of conjunctive queries**

Research goal: Understanding the complexity of PQE

What is the complexity of $\mathrm{PQE}(Q)$ depending on the query Q?

$\rightarrow$ Recall that we study **data complexity**, i.e., Q is **fixed** and the input is the **data**

For example:

- For $Q : R(x)$ the problem is **easy** (PTIME)
- For $Q : R(x), S(x, y), T(y)$ the problem is **hard** (#P-hard)

We will present the **dichotomy** of [Dalvi and Suciu, 2007, Dalvi and Suciu, 2012]:

- Small dichotomy: **conjunctive queries** that are **self-join-free** and **arity-two**
- Large dichotomy: arbitrary **unions of conjunctive queries**

Result of the form:

if Q *has a certain form* then $\mathrm{PQE}(Q)$ is in PTIME, otherwise it is #P-hard

The “small” Dalvi and Suciu dichotomy

- **Conjunctive query (CQ)**: existentially quantified conjunction of atoms

The “small” Dalvi and Suciu dichotomy

- Conjunctive query (CQ): existentially quantified conjunction of atoms
- Arity-two: all relations are binary
  - We represent the queries as graphs: $R(x, y), S(y, z)$ is $x \longrightarrow y \longrightarrow z$

The “small” Dalvi and Suciu dichotomy

- Conjunctive query (CQ): existentially quantified conjunction of atoms
- Arity-two: all relations are binary
  - We represent the queries as graphs: $R(x,y), S(y,z)$ is $x \longrightarrow y \longrightarrow z$
- Self-join-free CQ: only one edge of each color (no repeated color)

The “small” Dalvi and Suciu dichotomy

- **Conjunctive query (CQ)**: existentially quantified conjunction of atoms
- **Arity-two**: all relations are binary
  - We represent the queries as **graphs**: $R(x, y), S(y, z)$ is $x \xrightarrow{\quad\quad} y \xrightarrow{\quad\quad} z$
- **Self-join-free CQ**: only one edge of each color (no repeated color)

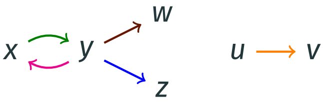
Theorem ([Dalvi and Suciu, 2007])

Let Q be an arity-two self-join-free CQ:

- *If Q is a **conjunction of stars**, then $\mathrm{PQE}(Q)$ is in **PTIME***
- *Otherwise, $\mathrm{PQE}(Q)$ is **#P-hard***

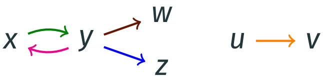
Conjunction of stars

- A **star** is a CQ with a **separator variable** that occurs in all edges
- A **conjunction of stars** is a conjunction of one or several stars



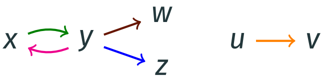
The following is **not a star**: $x \longrightarrow y \longrightarrow z \longrightarrow w$

Proving the small dichotomy (upper bound, 1)

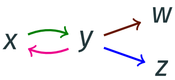


How to solve $\mathrm{PQE}(Q)$ for Q a conjunction of stars?

Proving the small dichotomy (upper bound, 1)



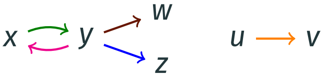
How to solve $\mathrm{PQE}(Q)$ for Q a conjunction of stars?



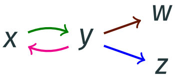
- We consider each connected component separately

→ Independent conjunction over the connected components

Proving the small dichotomy (upper bound, 1)



How to solve $\mathrm{PQE}(Q)$ for Q a conjunction of stars?



- We consider each connected component separately

→ Independent conjunction over the connected components

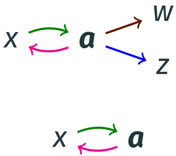
- We can test all possible values of the separator variable

→ Independent disjunction over the values of the separator

Proving the small dichotomy (upper bound, 2)

$$\begin{tikzcd} & & w \\ x \arrow[r, bend left] & a \arrow[l, bend left] \arrow[ur] \arrow[dr] & \\ & & z \end{tikzcd}$$

Proving the small dichotomy (upper bound, 2)



- For every match, we consider every **other variable** separately

→ **Independent conjunction** over the variables

Proving the small dichotomy (upper bound, 2)

$$\begin{tikzcd} & & w \\ x \arrow[r, bend left] & \boldsymbol{a} \arrow[l, bend left] \arrow[ru] \arrow[rd] & \\ & & z \end{tikzcd}$$

$$\begin{tikzcd} x \arrow[r, bend left] & \boldsymbol{a} \arrow[l, bend left] \end{tikzcd}$$

- For every match, we consider every other variable separately

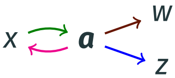
→ Independent conjunction over the variables

$$\begin{tikzcd} \boldsymbol{b_1} \arrow[r, bend left] & \boldsymbol{a} \arrow[l, bend left] \\ \boldsymbol{b_2} \arrow[r, bend left] & \boldsymbol{a} \arrow[l, bend left] \\ \boldsymbol{b_3} \arrow[r, bend left] & \boldsymbol{a} \arrow[l, bend left] \\ \vdots & \end{tikzcd}$$

- We consider every value for the other variable

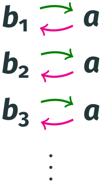
→ Independent disjunction over the possible assignments

Proving the small dichotomy (upper bound, 2)



- For every match, we consider every **other variable** separately

→ **Independent conjunction** over the variables



- We consider every value for the **other variable**

→ **Independent disjunction** over the possible assignments

- We consider every fact

→ **Independent conjunction** over the facts

→ Just **read the probability** of the ground fact $R(b, a)$.

Proving the small dichotomy (lower bound, 1)

Every arity-two self-join-free CQ which is not a conjunction of stars contains a pattern essentially like:

$$x \longrightarrow y \longrightarrow z \longrightarrow w$$

Proving the small dichotomy (lower bound, 1)

Every arity-two self-join-free CQ which is **not a conjunction of stars** contains a pattern essentially like:

$$x \longrightarrow y \longrightarrow z \longrightarrow w$$

We can **add facts with probability 1** to instances so the other facts are always satisfied, and focus on **only these three facts**

→ **Let us show #P-hardness of this query**

Proving the small dichotomy (lower bound, 2)

Let us show that $\mathrm{PQE}(Q)$ is **#P-hard** for the CQ Q : $x \longrightarrow y \longrightarrow z \longrightarrow w$

Proving the small dichotomy (lower bound, 2)

Let us show that $\mathrm{PQE}(Q)$ is **#P-hard** for the CQ Q : $x \longrightarrow y \longrightarrow z \longrightarrow w$

- Reduce from the problem of counting satisfying valuations of a Boolean formula
    - e.g., given $(x \lor y) \land z$, compute that it has 3 satisfying valuations

Proving the small dichotomy (lower bound, 2)

Let us show that $\mathrm{PQE}(Q)$ is **#P-hard** for the CQ Q : $x \xrightarrow{\quad} y \xrightarrow{\quad} z \xrightarrow{\quad} w$

- Reduce from the problem of **counting satisfying valuations** of a Boolean formula
    - e.g., given $(x \lor y) \land z$, compute that it has 3 satisfying valuations

- This problem is already **#P-hard** for so-called **PP2DNF formulas**:

Proving the small dichotomy (lower bound, 2)

Let us show that $\mathrm{PQE}(Q)$ is **#P-hard** for the CQ Q : $x \xrightarrow{\quad} y \xrightarrow{\quad} z \xrightarrow{\quad} w$

- Reduce from the problem of **counting satisfying valuations** of a Boolean formula
  - e.g., given $(x \lor y) \land z$, compute that it has 3 satisfying valuations
- This problem is already **#P-hard** for so-called **PP2DNF formulas**:
  - **Positive** (no negation) and **Partitioned variables**: $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$

Proving the small dichotomy (lower bound, 2)

Let us show that $\mathrm{PQE}(Q)$ is **#P-hard** for the CQ Q : $x \longrightarrow y \longrightarrow z \longrightarrow w$

- Reduce from the problem of **counting satisfying valuations** of a Boolean formula
  - e.g., given $(x \vee y) \wedge z$, compute that it has 3 satisfying valuations

- This problem is already **#P-hard** for so-called **PP2DNF formulas**:
  - **Positive** (no negation) and **Partitioned variables**: $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$
  - **2-DNF**: disjunction of clauses like $X_i \wedge Y_j$

Proving the small dichotomy (lower bound, 2)

Let us show that $\mathrm{PQE}(Q)$ is **#P-hard** for the CQ $Q: x \xrightarrow{} y \xrightarrow{} z \xrightarrow{} w$

- Reduce from the problem of **counting satisfying valuations** of a Boolean formula
  - e.g., given $(x \lor y) \land z$, compute that it has 3 satisfying valuations
- This problem is already **#P-hard** for so-called **PP2DNF formulas**:
  - **Positive** (no negation) and **Partitioned variables**: $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$
  - **2-DNF**: disjunction of clauses like $X_i \land Y_j$
- Example: $\phi : (X_1 \land Y_1) \lor (X_1 \land Y_2) \lor (X_2 \land Y_2) \lor (X_3 \land Y_1) \lor (X_3 \land Y_2)$

Proving the small dichotomy (lower bound, 3)

Reduce from #PP2DNF to $\mathrm{PQE}(Q)$ for CQ Q : $x \longrightarrow y \longrightarrow z \longrightarrow w$

Example: $\phi : (X_1 \wedge Y_1) \vee (X_1 \wedge Y_2) \vee (X_2 \wedge Y_2) \vee (X_3 \wedge Y_1) \vee (X_3 \wedge Y_2)$

Proving the small dichotomy (lower bound, 3)

Reduce from #PP2DNF to $\mathrm{PQE}(Q)$ for CQ Q : $x \longrightarrow y \longrightarrow z \longrightarrow w$

Example: $\phi : (X_1 \wedge Y_1) \vee (X_1 \wedge Y_2) \vee (X_2 \wedge Y_2) \vee (X_3 \wedge Y_1) \vee (X_3 \wedge Y_2)$

Build an instance I_ϕ from ϕ:

Proving the small dichotomy (lower bound, 3)

Reduce from **#PP2DNF** to $\mathrm{PQE}(Q)$ for CQ Q : $x \longrightarrow y \longrightarrow z \longrightarrow w$

Example: $\phi : (X_1 \wedge Y_1) \vee (X_1 \wedge Y_2) \vee (X_2 \wedge Y_2) \vee (X_3 \wedge Y_1) \vee (X_3 \wedge Y_2)$

Build an **instance** I_ϕ from ϕ:

$$\begin{tikzcd}
a'_1 \arrow[r, "1/2"] & a_1 \\
a'_2 \arrow[r, "1/2"] & a_2 \\
a'_3 \arrow[r, "1/2"] & a_3
\end{tikzcd}$$

Proving the small dichotomy (lower bound, 3)

Reduce from **#PP2DNF** to $\mathrm{PQE}(Q)$ for CQ Q : $x \longrightarrow y \longrightarrow z \longrightarrow w$

Example: $\phi : (X_1 \land Y_1) \lor (X_1 \land Y_2) \lor (X_2 \land Y_2) \lor (X_3 \land Y_1) \lor (X_3 \land Y_2)$

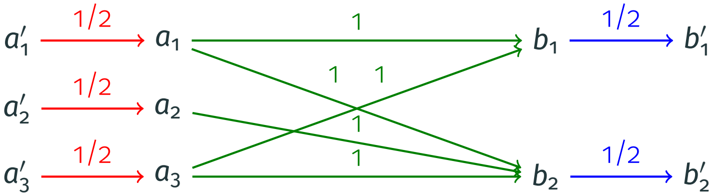
Build an **instance** I_ϕ from ϕ:

Proving the small dichotomy (lower bound, 3)

Reduce from #PP2DNF to $\mathrm{PQE}(Q)$ for CQ Q : $x \longrightarrow y \longrightarrow z \longrightarrow w$

Example: $\phi : (X_1 \land Y_1) \lor (X_1 \land Y_2) \lor (X_2 \land Y_2) \lor (X_3 \land Y_1) \lor (X_3 \land Y_2)$

Build an instance I_ϕ from ϕ:

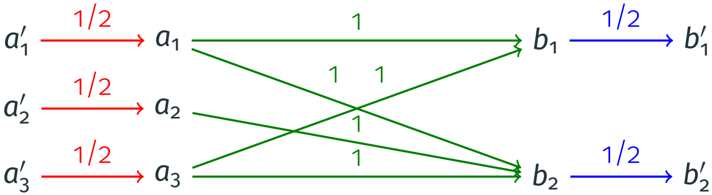


Proving the small dichotomy (lower bound, 3)

Reduce from **#PP2DNF** to $\mathrm{PQE}(Q)$ for CQ Q : $x \xrightarrow{\quad} y \xrightarrow{\quad} z \xrightarrow{\quad} w$

Example: $\phi : (X_1 \wedge Y_1) \vee (X_1 \wedge Y_2) \vee (X_2 \wedge Y_2) \vee (X_3 \wedge Y_1) \vee (X_3 \wedge Y_2)$

Build an **instance** I_ϕ from ϕ:



Idea:

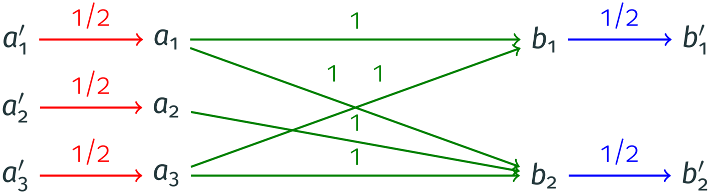
- **Valuations** of ϕ correspond to **possible worlds** of I_ϕ

Proving the small dichotomy (lower bound, 3)

Reduce from **#PP2DNF** to $\mathrm{PQE}(Q)$ for CQ Q : $x \longrightarrow y \longrightarrow z \longrightarrow w$

Example: $\phi : (X_1 \wedge Y_1) \vee (X_1 \wedge Y_2) \vee (X_2 \wedge Y_2) \vee (X_3 \wedge Y_1) \vee (X_3 \wedge Y_2)$

Build an **instance** I_ϕ from ϕ:



Idea:

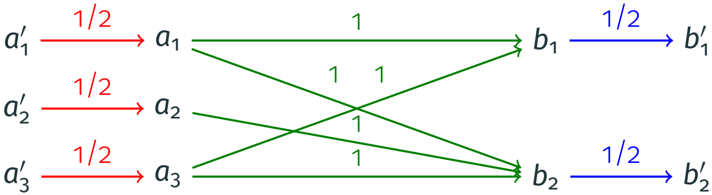
- **Valuations** of ϕ correspond to **possible worlds** of I_ϕ
- A valuation **satisfies** ϕ iff the corresponding possible world **satisfies** Q

Proving the small dichotomy (lower bound, 3)

Reduce from **#PP2DNF** to $\mathrm{PQE}(Q)$ for CQ Q : $x \longrightarrow y \longrightarrow z \longrightarrow w$

Example: $\phi : (X_1 \land Y_1) \lor (X_1 \land Y_2) \lor (X_2 \land Y_2) \lor (X_3 \land Y_1) \lor (X_3 \land Y_2)$

Build an **instance** I_ϕ from ϕ:



Idea:

- **Valuations** of ϕ correspond to **possible worlds** of I_ϕ
- A valuation **satisfies** ϕ iff the corresponding possible world **satisfies** Q

→ The **probability** of Q on I_ϕ is the **number of accepting valuations of** ϕ, divided by the number of valuations ($2^{-|\mathrm{Vars}|}$)

Extending beyond arity-two (1)

How can we extend beyond arity-two queries?

Theorem ([Dalvi and Suciu, 2007])

Let Q be a ~~arity-two~~ self-join-free CQ:

- *If Q is ~~a conjunction of stars~~ hierarchical, then $\mathrm{PQE}(Q)$ is in PTIME*
- *Otherwise, $\mathrm{PQE}(Q)$ is #P-hard*

Extending beyond arity-two (2)

Class of Hierarchical CQs defined inductively:

- A query with no variables is hierarchical

Extending beyond arity-two (2)

Class of **Hierarchical** CQs defined inductively:

- A query with **no variables** is hierarchical
- A **conjunction** of hierarchical connected components is hierarchical

Extending beyond arity-two (2)

Class of Hierarchical CQs defined inductively:

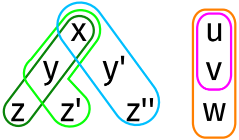
- A query with no variables is hierarchical
- A conjunction of hierarchical connected components is hierarchical
- Induction case: for a connected CQ:
    - It must have a separator variable occurring in all atoms
    - If we remove this separator variable, the query must be hierarchical

Extending beyond arity-two (2)

Class of **Hierarchical** CQs defined inductively:

- A query with **no variables** is hierarchical
- A **conjunction** of hierarchical connected components is hierarchical
- Induction case: for a connected CQ:
  - It must have a **separator variable** occurring in all atoms
  - If we remove this separator variable, the query must be hierarchical

$$\exists x\,(\exists y\,(\exists z\; R_1(x,y,z)) \land (\exists z'\; R_2(x,y,z'))) \land (\exists y' \exists z''\; R_3(x,y',z''))$$
$$\land\,(\exists u\,(\exists v\; R_4(u,v)) \land (\exists w\; R_5(u,v,w)))$$



Extending beyond arity-two (3)

How does the proof change?

Extending beyond arity-two (3)

How does the proof change?

- **Upper bound**: we can generalize the algorithm
  - **Independent AND** of connected components
  - **Independent OR** of possible choices for the separator variable
  - **Both cases use self-join-freeness!**

Extending beyond arity-two (3)

How does the proof change?

- **Upper bound**: we can generalize the algorithm
  - **Independent AND** of connected components
  - **Independent OR** of possible choices for the separator variable
  - **Both cases use self-join-freeness!**
- **Lower bound**: a non-hierarchical expression contains a pattern like

$$x \longrightarrow y \longrightarrow z \longrightarrow w$$

Extending beyond arity-two (3)

How does the proof change?

- **Upper bound**: we can generalize the algorithm
  - **Independent AND** of connected components
  - **Independent OR** of possible choices for the separator variable
  - **Both cases use self-join-freeness!**
- **Lower bound**: a non-hierarchical expression contains a pattern like

$$x \longrightarrow y \longrightarrow z \longrightarrow w$$

Via **equivalent characterization**: a non-hierarchical query has two variables x and y and:

- One atom containing **x and y**
- One atom containing **x but not y**
- One atom containing **y but not x**

The “big” Dalvi and Suciu dichotomy

Full dichotomy on the **unions of conjunctive queries** (UCQs):

Theorem ([Dalvi and Suciu, 2012])

Let Q *be a UCQ:*

- *If* Q *is handled by a complicated algorithm then* $\mathrm{PQE}(Q)$ *is in* ***PTIME***
- *Otherwise,* $\mathrm{PQE}(Q)$ *is* ***#P-hard***

The "big" Dalvi and Suciu dichotomy

Full dichotomy on the **unions of conjunctive queries** (UCQs):

Theorem ([Dalvi and Suciu, 2012])

Let Q be a UCQ:

- *If Q is handled by a complicated algorithm then $\mathrm{PQE}(Q)$ is in* ***PTIME***
- *Otherwise, $\mathrm{PQE}(Q)$ is* ***#P-hard***

This result is **far more challenging**:

- **Upper bound:**
    - an algorithm generalizing the previous case with **inclusion-exclusion**
    - many **unpleasant details** (e.g., a ranking transformation)

The "big" Dalvi and Suciu dichotomy

Full dichotomy on the unions of conjunctive queries (UCQs):

Theorem ([Dalvi and Suciu, 2012])

Let Q be a UCQ:

- *If Q is handled by a complicated algorithm then $\mathrm{PQE}(Q)$ is in PTIME*
- *Otherwise, $\mathrm{PQE}(Q)$ is #P-hard*

This result is far more challenging:

- Upper bound:
  - an algorithm generalizing the previous case with inclusion-exclusion
  - many unpleasant details (e.g., a ranking transformation)
- Lower bound: hardness proof on minimal cases where the algorithm does not work (very challenging)

References i

Dalvi, N. and Suciu, D. (2007).
The dichotomy of conjunctive queries on probabilistic structures.
In *Proc. PODS*.

Dalvi, N. and Suciu, D. (2012).
The dichotomy of probabilistic inference for unions of conjunctive queries.
J. ACM, 59(6).

Probabilistic Databases: Provenance Circuits and Knowledge Compilation

EDBT-Intended Summer School

Antoine Amarilli

Recall: Boolean Provenance

- Relational database instance I: set of facts
- Boolean query Q: take an instance and answer yes/no

Recall: Boolean Provenance

- **Relational database instance** I: set of facts
- **Boolean query** Q: take an instance and answer yes/no

Example: query Q:

$\exists xyz\ R(x, y) \wedge S(y, z)$

Recall: Boolean Provenance

- **Relational database instance** I: set of facts
- **Boolean query** Q: take an instance and answer yes/no

Example: query Q:

$\exists xyz\ R(x, y) \wedge S(y, z)$

R	
a	b
a'	b

S	
b	c

Recall: Boolean Provenance

- **Relational database instance** I: set of facts
- **Boolean query** Q: take an instance and answer yes/no
- **Boolean provenance** of Q on I: a **Boolean circuit** over the facts of I accepting exactly the **subsets** of I where Q is true

Example: query Q:

$\exists xyz\ R(x, y) \land S(y, z)$

R	
a	b
a'	b

S	
b	c

Recall: Boolean Provenance

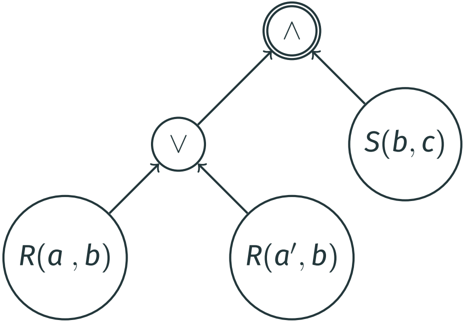
- **Relational database instance** I: set of facts
- **Boolean query** Q: take an instance and answer yes/no
- **Boolean provenance** of Q on I: a **Boolean circuit** over the facts of I accepting exactly the **subsets** of I where Q is true

Example: query Q:

$\exists xyz\, R(x, y) \land S(y, z)$

R	
a	b
a'	b

S	
b	c



Related work: Semiring provenance

Semiring provenance ([Green et al., 2007], on Tuesday): annotate results of a **relational algebra** query with a **semiring expression**

A B C	
a b c	p
d b e	r
f g e	s

(a)

A C	
a c	$\{p\}$
a e	$\{p, r\}$
d c	$\{p, r\}$
d e	$\{r, s\}$
f e	$\{r, s\}$

(b)

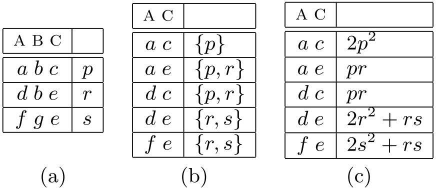
A C	
a c	$2p^2$
a e	pr
d c	pr
d e	$2r^2 + rs$
f e	$2s^2 + rs$

(c)

Figure 5: Why-prov. and provenance polynomials

Related work: Semiring provenance

Semiring provenance ([Green et al., 2007], on Tuesday): annotate results of a relational algebra query with a semiring expression



A	B	C	
a	b	c	p
d	b	e	r
f	g	e	s

(a)

A	C	
a	c	$\{p\}$
a	e	$\{p, r\}$
d	c	$\{p, r\}$
d	e	$\{r, s\}$
f	e	$\{r, s\}$

(b)

A	C	
a	c	$2p^2$
a	e	pr
d	c	pr
d	e	$2r^2 + rs$
f	e	$2s^2 + rs$

(c)

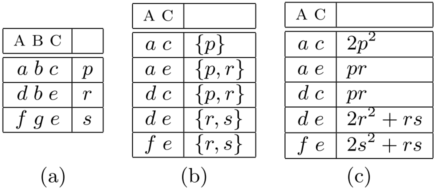
Figure 5: Why-prov. and provenance polynomials

What is the difference?

- We only care about **Boolean provenance**

Related work: Semiring provenance

Semiring provenance ([Green et al., 2007], on Tuesday): annotate results of a relational algebra query with a semiring expression



(a)

A	B	C	
a	b	c	p
d	b	e	r
f	g	e	s

(b)

A	C	
a	c	$\{p\}$
a	e	$\{p, r\}$
d	c	$\{p, r\}$
d	e	$\{r, s\}$
f	e	$\{r, s\}$

(c)

A	C	
a	c	$2p^2$
a	e	pr
d	c	pr
d	e	$2r^2 + rs$
f	e	$2s^2 + rs$

Figure 5: Why-prov. and provenance polynomials

What is the difference?

- We only care about **Boolean provenance**
  → No **multiplicity** of facts or derivations
- **Circuit representation:** more concise

The intensional approach to PQE

- **Previously**, for a tractable query Q: we can solve $\mathrm{PQE}(Q)$
- Now, let's see the **intensional approach**
  - Compute a **circuit** representing the **Boolean provenance** of Q
  - For **tractable Q** the circuit falls in a **tractable class** and we can compute the probability

The intensional approach to PQE

- **Previously**, for a tractable query Q: we can solve $\mathrm{PQE}(Q)$
- Now, let's see the **intensional approach**
  - Compute a **circuit** representing the **Boolean provenance** of Q
  - For **tractable Q** the circuit falls in a **tractable class** and we can compute the probability
- Why do that?
  - More **modular**, no **numerical computations**, connect to known **circuit classes**
  - **Knowledge compilation**: use circuits for **other tasks**, e.g., provenance, enumeration...

The intensional approach to PQE

- **Previously**, for a tractable query Q: we can solve $\mathrm{PQE}(Q)$
- Now, let's see the **intensional approach**
  - Compute a **circuit** representing the **Boolean provenance** of Q
  - For **tractable Q** the circuit falls in a **tractable class** and we can compute the probability
- Why do that?
  - More **modular**, no **numerical computations**, connect to known **circuit classes**
  - **Knowledge compilation**: use circuits for **other tasks**, e.g., provenance, enumeration...

Without knowledge compilation:

$O(n^2)$ algorithms

Setting A → Task 1

Setting A → Task 2

Setting B → Task 1

Setting B → Task 2

The intensional approach to PQE

- **Previously**, for a tractable query Q: we can solve $\mathrm{PQE}(Q)$
- Now, let's see the **intensional approach**
  - Compute a **circuit** representing the **Boolean provenance** of Q
  - For **tractable Q** the circuit falls in a **tractable class** and we can compute the probability
- Why do that?
  - More **modular**, no **numerical computations**, connect to known **circuit classes**
  - **Knowledge compilation**: use circuits for **other tasks**, e.g., provenance, enumeration...

Without knowledge compilation:

$O(n^2)$ algorithms

With knowledge compilation:

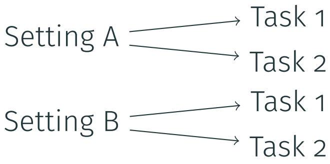
$O(n)$ algorithms

The intensional approach to PQE

- **Previously**, for a tractable query Q: we can solve $\mathrm{PQE}(Q)$
- Now, let's see the **intensional approach**
  - Compute a **circuit** representing the **Boolean provenance** of Q
  - For **tractable Q** the circuit falls in a **tractable class** and we can compute the probability
- Why do that?
  - More **modular**, no **numerical computations**, connect to known **circuit classes**
  - **Knowledge compilation**: use circuits for **other tasks**, e.g., provenance, enumeration...

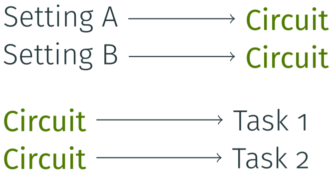
Without knowledge compilation:

$O(n^2)$ algorithms


Setting A → Task 1
Setting A → Task 2
Setting B → Task 1
Setting B → Task 2

With knowledge compilation:

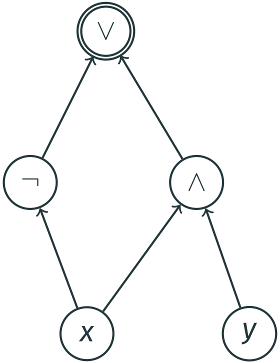
$O(n)$ algorithms


Setting A → **Circuit**
Setting B → **Circuit**

Circuit → Task 1
Circuit → Task 2

Boolean circuit representations

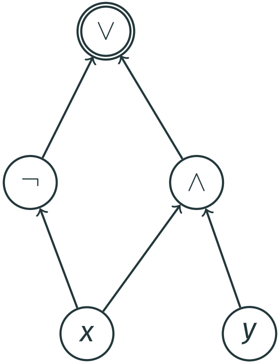
Circuits are just a way to represent **Boolean formulas**
while factoring common subexpressions (more concise)



- Directed acyclic graph of **gates**
- **Output** gate: ◎
- **Variable** gates: x
- **Internal** gates: $\vee$ $\wedge$ $\neg$

Boolean circuit representations

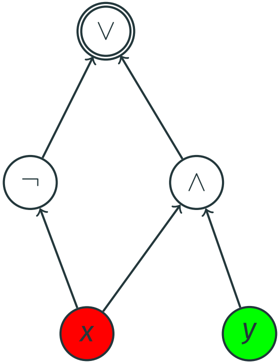
Circuits are just a way to represent **Boolean formulas**
while factoring common subexpressions (more concise)



- Directed acyclic graph of **gates**
- **Output** gate: ◎
- **Variable** gates: ⓧ
- **Internal** gates: ∨ ∧ ¬
- **Valuation**: function from variables to $\{0, 1\}$
  Example: $\nu = \{x \mapsto 0,\ y \mapsto 1\}$...

Boolean circuit representations

Circuits are just a way to represent **Boolean formulas**
while factoring common subexpressions (more concise)

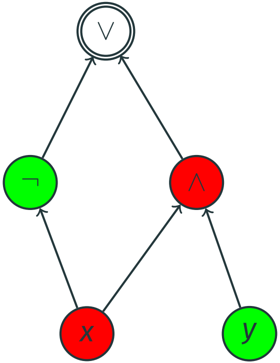


- Directed acyclic graph of **gates**
- **Output** gate: ◎
- **Variable** gates: ⓧ
- **Internal** gates: $\vee$ $\wedge$ $\neg$

- **Valuation**: function from variables to $\{0, 1\}$
  Example: $\nu = \{x \mapsto 0,\ y \mapsto 1\}$...

Boolean circuit representations

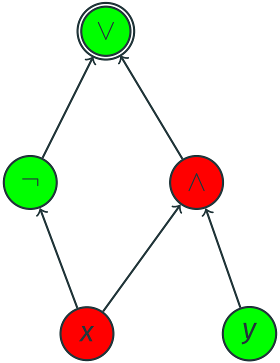
Circuits are just a way to represent **Boolean formulas**
while factoring common subexpressions (more concise)



- Directed acyclic graph of **gates**
- **Output** gate: ◎
- **Variable** gates: x
- **Internal** gates: $\vee$ $\wedge$ $\neg$
- **Valuation**: function from variables to $\{0, 1\}$
  Example: $\nu = \{x \mapsto 0,\ y \mapsto 1\}$...

Boolean circuit representations

Circuits are just a way to represent **Boolean formulas**
while factoring common subexpressions (more concise)



- Directed acyclic graph of **gates**
- **Output** gate: ◎
- **Variable** gates: x
- **Internal** gates: $\lor$ $\land$ $\neg$
- **Valuation**: function from variables to $\{0, 1\}$
  Example: $\nu = \{x \mapsto 0,\ y \mapsto 1\}$... mapped to 1

Computing Boolean provenance: theory

- Unions of Conjunctive Queries (UCQ)

Theorem

For any UCQ, given an instance, we can construct its Boolean provenance in PTIME

Computing Boolean provenance: theory

- Unions of Conjunctive Queries (UCQ)

Theorem

For any UCQ, given an instance, we can construct its Boolean provenance in PTIME (disjunction of all matches)

Computing Boolean provenance: theory

- Unions of Conjunctive Queries (UCQ)

Theorem

For any UCQ, given an instance, we can construct its Boolean provenance in PTIME (disjunction of all matches)

- Acyclic Conjunctive Queries (ACQ)

Theorem

For any ACQ, given an instance, we can construct its Boolean provenance in linear time

Computing Boolean provenance: theory

- Unions of Conjunctive Queries (UCQ)

Theorem

For any UCQ, given an instance, we can construct its Boolean provenance in PTIME (disjunction of all matches)

- Acyclic Conjunctive Queries (ACQ)

Theorem

For any ACQ, given an instance, we can construct its Boolean provenance in linear time (following a join tree)

Computing Boolean provenance: theory

- Unions of Conjunctive Queries (UCQ)

Theorem

For any UCQ, given an instance, we can construct its Boolean provenance in PTIME
(disjunction of all matches)

- Acyclic Conjunctive Queries (ACQ)

Theorem

For any ACQ, given an instance, we can construct its Boolean provenance in linear time
(following a join tree)

- Regular path queries (RPQ), Datalog, etc.

Theorem [Deutch et al., 2014]

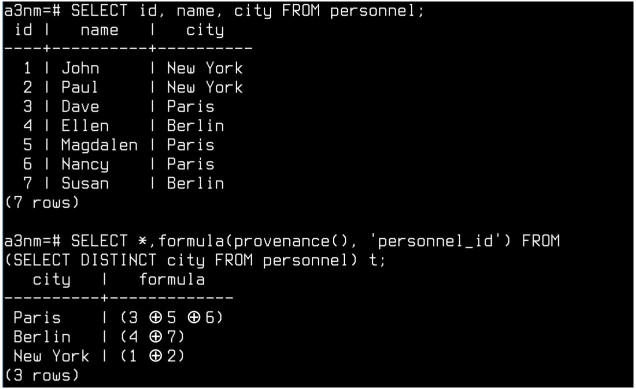
For any Datalog query, given an instance, we can get its Boolean provenance in PTIME

Computing Boolean provenance: practice

- **ProvSQL:** PostgreSQL extension to compute provenance
- Keeps track of the **provenance** of query results as a **circuit**

Computing Boolean provenance: practice

- **ProvSQL:** PostgreSQL extension to compute provenance
- Keeps track of the **provenance** of query results as a **circuit**



```
a3nm=# SELECT id, name, city FROM personnel;
 id |   name   |   city
----+----------+----------
  1 | John     | New York
  2 | Paul     | New York
  3 | Dave     | Paris
  4 | Ellen    | Berlin
  5 | Magdalen | Paris
  6 | Nancy    | Paris
  7 | Susan    | Berlin
(7 rows)

a3nm=# SELECT *,formula(provenance(), 'personnel_id') FROM
(SELECT DISTINCT city FROM personnel) t;
   city   |   formula
----------+-------------
 Paris    | (3 ⊕ 5 ⊕ 6)
 Berlin   | (4 ⊕ 7)
 New York | (1 ⊕ 2)
(3 rows)
```

You can run it! `https://github.com/PierreSenellart/provsql`

Summary: Boolean provenance for PQE

- We have fixed the Boolean query Q

Summary: Boolean provenance for PQE

- We have fixed the Boolean query Q

Example: query Q:

$\exists xyz\ R(x,y) \wedge S(y,z)$

Summary: Boolean provenance for PQE

- We have fixed the **Boolean query** Q
- We are given an input **TID** I with a probability P of each fact

Example: query Q:

$\exists xyz\ R(x, y) \land S(y, z)$

Summary: Boolean provenance for PQE

- We have fixed the **Boolean query** Q
- We are given an input **TID** I with a probability P of each fact

Example: query Q:

$\exists xyz\ R(x, y) \wedge S(y, z)$

R	
a	b
a'	b

S	
b	c

Summary: Boolean provenance for PQE

- We have fixed the **Boolean query** Q
- We are given an input **TID** I with a probability P of each fact

Example: query Q:

$\exists xyz\ R(x,y) \land S(y,z)$

R		
a	b	80%
a'	b	90%

S		
b	c	40%

Summary: Boolean provenance for PQE

- We have fixed the **Boolean query** Q
- We are given an input **TID** I with a probability P of each fact
- We have computed a **Boolean provenance circuit** of Q on I

Example: query Q:

$\exists xyz\ R(x, y) \wedge S(y, z)$

R		
a	b	80%
a'	b	90%

S		
b	c	40%

Summary: Boolean provenance for PQE

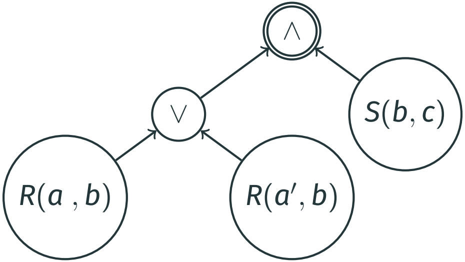
- We have fixed the Boolean query Q
- We are given an input TID I with a probability P of each fact
- We have computed a Boolean provenance circuit of Q on I

Example: query Q:

$\exists xyz\, R(x, y) \land S(y, z)$

R		
a	b	80%
a'	b	90%

S		
b	c	40%



Summary: Boolean provenance for PQE

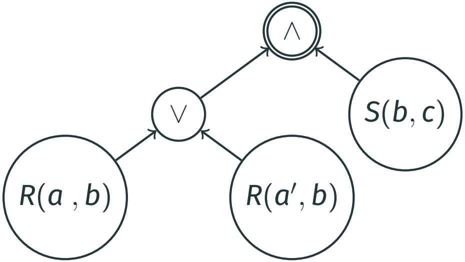
- We have fixed the **Boolean query** Q
- We are given an input **TID** I with a probability P of each fact
- We have computed a **Boolean provenance circuit** of Q on I
- Each **variable of the circuit** (fact of the database) has an **independent probability**

Example: query Q:

$\exists xyz\ R(x,y) \wedge S(y,z)$

R		
a	b	80%
a'	b	90%

S		
b	c	40%



Summary: Boolean provenance for PQE

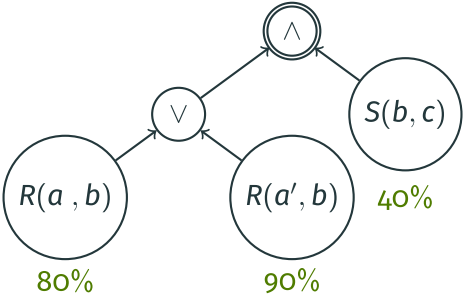
- We have fixed the **Boolean query** Q
- We are given an input **TID** I with a probability P of each fact
- We have computed a **Boolean provenance circuit** of Q on I
- Each **variable of the circuit** (fact of the database) has an **independent probability**

Example: query Q:

$\exists xyz\ R(x,y) \land S(y,z)$

R		
a	b	80%
a'	b	90%

S		
b	c	40%



Summary: Boolean provenance for PQE

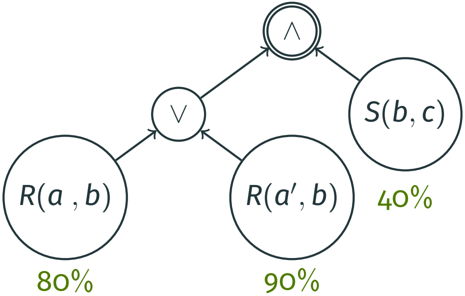
- We have fixed the **Boolean query** Q
- We are given an input **TID** I with a probability P of each fact
- We have computed a **Boolean provenance circuit** of Q on I
- Each **variable of the circuit** (fact of the database) has an **independent probability**
- Each **Boolean valuation** of the circuit corresponds to a **possible world** J of I and the circuit evaluates to **true** iff...

Example: query Q:

$\exists xyz\ R(x, y) \wedge S(y, z)$

R		
a	b	80%
a'	b	90%

S		
b	c	40%



Summary: Boolean provenance for PQE

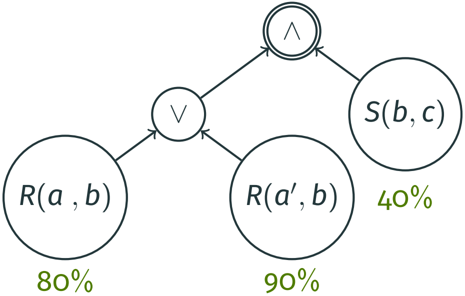
- We have fixed the **Boolean query** Q
- We are given an input **TID** I with a probability P of each fact
- We have computed a **Boolean provenance circuit** of Q on I
- Each **variable of the circuit** (fact of the database) has an **independent probability**
- Each **Boolean valuation** of the circuit corresponds to a **possible world** J of I and the circuit evaluates to **true** iff... **J satisfies Q**

Example: query Q:

$\exists xyz\ R(x, y) \wedge S(y, z)$

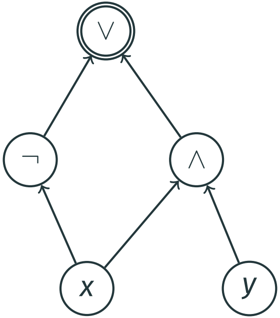
R		
a	b	80%
a'	b	90%

S		
b	c	40%



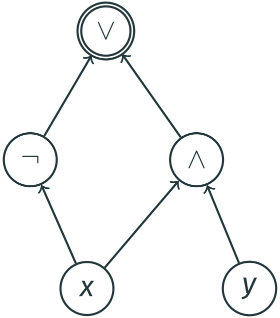
Computing the probability of the circuit

- We now have a Boolean provenance circuit over the database facts



Computing the probability of the circuit

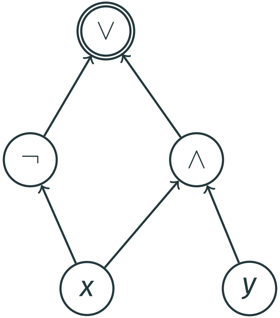
- We now have a Boolean provenance circuit over the database facts



- $P(x) = 40\%$
- $P(y) = 50\%$

Computing the probability of the circuit

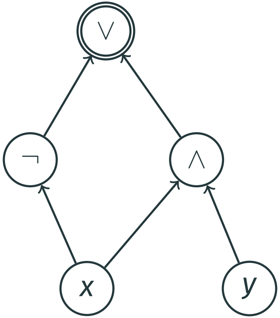
- We now have a Boolean provenance circuit over the database facts
- Each variable x is true independently with probability $P(x)$ (probability of the fact)



- $P(x) = 40\%$
- $P(y) = 50\%$

Computing the probability of the circuit

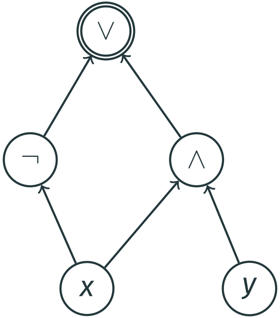
- We now have a Boolean provenance circuit over the database facts
- Each variable x is true independently with probability $P(x)$ (probability of the fact)
- What is the probability that the circuit evaluates to true?



- $P(x) = 40\%$
- $P(y) = 50\%$

Computing the probability of the circuit

- We now have a Boolean provenance circuit over the database facts
- Each variable x is true independently with probability $P(x)$ (probability of the fact)
- What is the probability that the circuit evaluates to true?

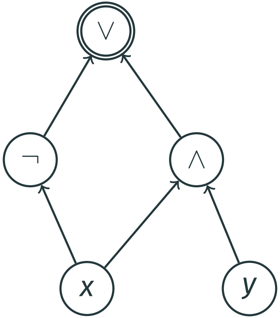


- In general, #P-hard (harder than SAT)

- $P(x) = 40\%$
- $P(y) = 50\%$

Computing the probability of the circuit

- We now have a **Boolean provenance circuit** over the **database facts**
- Each variable x is true **independently** with probability $P(x)$ (probability of the fact)
- What is the probability that the circuit **evaluates to true**?

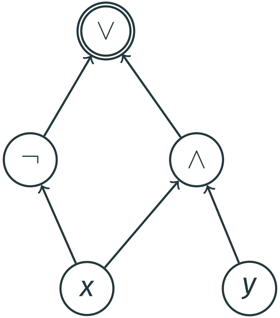


- $P(x) = 40\%$
- $P(y) = 50\%$

- In general, **#P-hard** (harder than SAT)
- Here it's **easy**:

Computing the probability of the circuit

- We now have a **Boolean provenance circuit** over the **database facts**
- Each variable x is true **independently** with probability $P(x)$ (probability of the fact)
- What is the probability that the circuit **evaluates to true**?

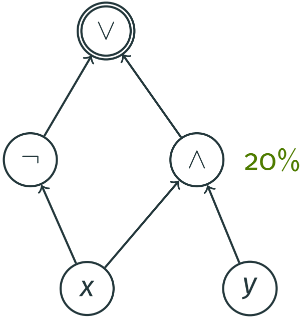


- $P(x) = 40\%$
- $P(y) = 50\%$

- In general, **#P-hard** (harder than SAT)
- Here it's **easy**:
  - The inputs to the **$\wedge$-gate** are **independent**

Computing the probability of the circuit

- We now have a Boolean provenance circuit over the database facts
- Each variable x is true independently with probability $P(x)$ (probability of the fact)
- What is the probability that the circuit evaluates to true?

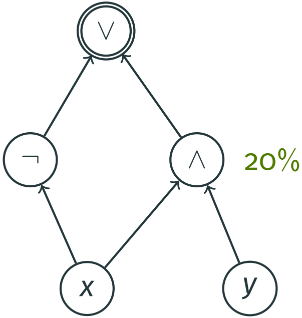


- $P(x) = 40\%$
- $P(y) = 50\%$

- In general, #P-hard (harder than SAT)
- Here it's easy:
  - The inputs to the $\wedge$-gate are independent

Computing the probability of the circuit

- We now have a **Boolean provenance circuit** over the **database facts**
- Each variable x is true **independently** with probability $P(x)$ (probability of the fact)
- What is the probability that the circuit **evaluates to true**?

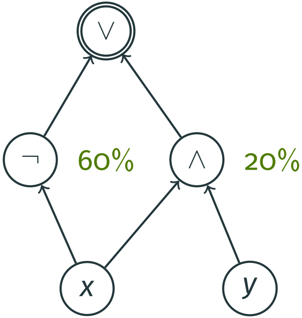


- $P(x) = 40\%$
- $P(y) = 50\%$

- In general, **#P-hard** (harder than SAT)
- Here it's **easy**:
  - The inputs to the **$\wedge$-gate** are **independent**
  - The **$\neg$-gate** has probability $1 - P(\text{input})$

Computing the probability of the circuit

- We now have a **Boolean provenance circuit** over the **database facts**
- Each variable x is true **independently** with probability $P(x)$ (probability of the fact)
- What is the probability that the circuit **evaluates to true**?

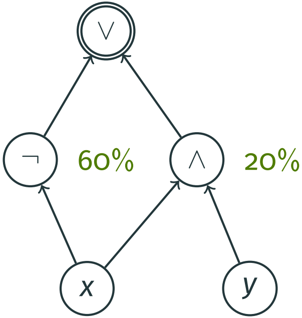


- $P(x) = 40\%$
- $P(y) = 50\%$

- In general, **#P-hard** (harder than SAT)
- Here it's **easy**:
  - The inputs to the **$\wedge$-gate** are **independent**
  - The **$\neg$-gate** has probability $1 - P(\text{input})$

Computing the probability of the circuit

- We now have a **Boolean provenance circuit** over the **database facts**
- Each variable x is true **independently** with probability $P(x)$ (probability of the fact)
- What is the probability that the circuit **evaluates to true**?

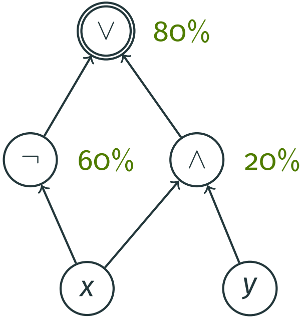


- $P(x) = 40\%$
- $P(y) = 50\%$

- In general, **#P-hard** (harder than SAT)
- Here it's **easy**:
  - The inputs to the **∧-gate** are **independent**
  - The **¬-gate** has probability $1 - P(\text{input})$
  - The **∨-gate** has **mutually exclusive** inputs

Computing the probability of the circuit

- We now have a **Boolean provenance circuit** over the **database facts**
- Each variable x is true **independently** with probability $P(x)$ (probability of the fact)
- What is the probability that the circuit **evaluates to true**?

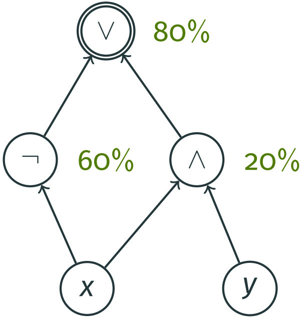


- $P(x) = 40\%$
- $P(y) = 50\%$

- In general, **#P-hard** (harder than SAT)
- Here it's **easy**:
  - The inputs to the $\wedge$**-gate** are **independent**
  - The $\neg$**-gate** has probability $1 - P(\text{input})$
  - The $\vee$**-gate** has **mutually exclusive** inputs

Computing the probability of the circuit

- We now have a **Boolean provenance circuit** over the **database facts**
- Each variable x is true **independently** with probability $P(x)$ (probability of the fact)
- What is the probability that the circuit **evaluates to true**?



- $P(x) = 40\%$
- $P(y) = 50\%$

- In general, **#P-hard** (harder than SAT)
- Here it's **easy**:
  - The inputs to the $\wedge$**-gate** are **independent**
  - The $\neg$**-gate** has probability $1 - P(\text{input})$
  - The $\vee$**-gate** has **mutually exclusive** inputs

→ The circuit that we constructed falls in a **restricted class** satisfying such conditions

A tractable circuit class: d-DNNFs

d-DNNF requirements

A tractable circuit class: d-DNNFs

d-DNNF requirements

- ¬ gates only have variables as inputs

A tractable circuit class: d-DNNFs

d-DNNF requirements

- ¬ gates only have **variables** as inputs
- ∨ gates always have **mutually exclusive** inputs

A tractable circuit class: d-DNNFs

d-DNNF requirements

- $\neg$ gates only have **variables** as inputs
- $\vee$ gates always have **mutually exclusive** inputs
- $\wedge$ gates are all on **independent** inputs

A tractable circuit class: d-DNNFs

d-DNNF requirements

… make probability computation easy!

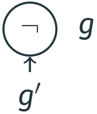
- ¬ gates only have **variables** as inputs
- ∨ gates always have **mutually exclusive** inputs
- ∧ gates are all on **independent** inputs

A tractable circuit class: d-DNNFs

d-DNNF requirements

- $\neg$ gates only have **variables** as inputs
- $\vee$ gates always have **mutually exclusive** inputs
- $\wedge$ gates are all on **independent** inputs

… make probability computation **easy**!

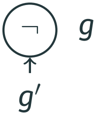


A tractable circuit class: d-DNNFs

d-DNNF requirements

- $\neg$ gates only have **variables** as inputs
- $\vee$ gates always have **mutually exclusive** inputs
- $\wedge$ gates are all on **independent** inputs

… make probability computation **easy**!



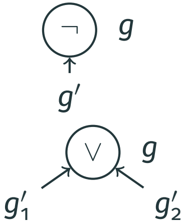
$$P(g) := 1 - P(g')$$

A tractable circuit class: d-DNNFs

d-DNNF requirements

- $\neg$ gates only have **variables** as inputs
- $\vee$ gates always have **mutually exclusive** inputs
- $\wedge$ gates are all on **independent** inputs

… make probability computation **easy**!



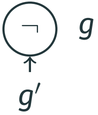
$$P(g) := 1 - P(g')$$

A tractable circuit class: d-DNNFs

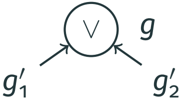
d-DNNF requirements

- $\neg$ gates only have **variables** as inputs
- $\vee$ gates always have **mutually exclusive** inputs
- $\wedge$ gates are all on **independent** inputs

… make probability computation **easy**!



$$P(g) := 1 - P(g')$$



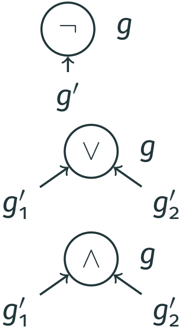
$$P(g) := P(g'_1) + P(g'_2)$$

A tractable circuit class: d-DNNFs

d-DNNF requirements

- $\neg$ gates only have **variables** as inputs
- $\vee$ gates always have **mutually exclusive** inputs
- $\wedge$ gates are all on **independent** inputs

… make probability computation **easy**!



$$P(g) := 1 - P(g')$$

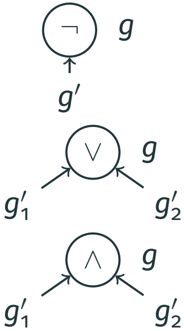
$$P(g) := P(g'_1) + P(g'_2)$$

A tractable circuit class: d-DNNFs

d-DNNF requirements

- ¬ gates only have **variables** as inputs
- ∨ gates always have **mutually exclusive** inputs
- ∧ gates are all on **independent** inputs

… make probability computation **easy**!



$$P(g) := 1 - P(g')$$

$$P(g) := P(g'_1) + P(g'_2)$$

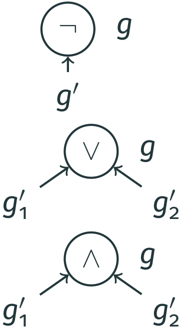
$$P(g) := P(g'_1) \times P(g'_2)$$

A tractable circuit class: d-DNNFs

d-DNNF requirements

- $\neg$ gates only have **variables** as inputs
- $\vee$ gates always have **mutually exclusive** inputs
- $\wedge$ gates are all on **independent** inputs

… make probability computation **easy**!



$$P(g) := 1 - P(g')$$

$$P(g) := P(g'_1) + P(g'_2)$$

$$P(g) := P(g'_1) \times P(g'_2)$$

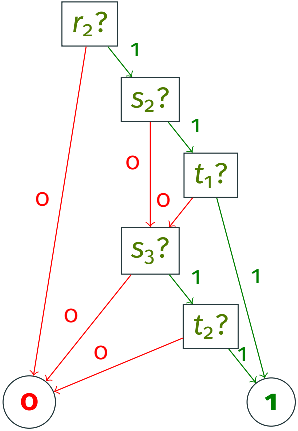
→ d-DNNFs are one of many **tractable circuit classes** in **knowledge compilation**

Other tractable circuit classes

- **Read-once formula:** Boolean formula where each variable occurs at most once
  → If the Boolean provenance is written in this way, we can compute the probability with independent AND, independent OR, negation

Other tractable circuit classes

- **Read-once formula:** Boolean formula where each variable occurs at most once
  - → If the Boolean provenance is written in this way, we can compute the probability with independent AND, independent OR, negation
- **Binary decision diagram**, e.g., **OBDDs**

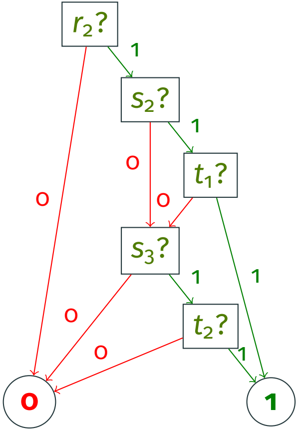


Other tractable circuit classes

- **Read-once formula:** Boolean formula where each variable occurs at most once
  → If the Boolean provenance is written in this way, we can compute the probability with independent AND, independent OR, negation

- **Binary decision diagram**, e.g., **OBDDs**

- Restricted classes of **d-DNNF**:
  - **dec-DNNF:** disjunction gates are of the form $x \land \alpha \lor \neg x \land \beta$
  - **d-SDNNF:** the circuit uses variables in a **structured** way



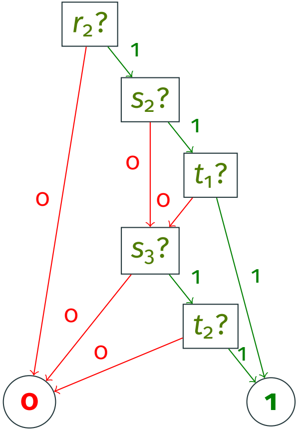
Other tractable circuit classes

- **Read-once formula:** Boolean formula where each variable occurs at most once
  → If the Boolean provenance is written in this way, we can compute the probability with independent AND, independent OR, negation

- **Binary decision diagram**, e.g., **OBDDs**

- Restricted classes of **d-DNNF**:
  - **dec-DNNF:** disjunction gates are of the form $x \land \alpha \lor \neg x \land \beta$
  - **d-SDNNF:** the circuit uses variables in a **structured** way

$$\begin{tikzcd}
& \text{dec-DNNF} \arrow[rr, two heads] & & \text{d-DNNF} \arrow[rr, two heads] & & \text{DNNF} \\
\text{dec-SDNNF} \arrow[ur, two heads] \arrow[rr, two heads] & & \text{d-SDNNF} \arrow[ur, two heads] \arrow[rr, two heads] & & \text{SDNNF} \arrow[ur, two heads] & \\
& \text{FBDD} \arrow[uu] \arrow[rr, two heads] & & \text{uFBDD} \arrow[uu] \arrow[rr, two heads] & & \text{nFBDD} \arrow[uu] \\
\text{OBDD} \arrow[uu] \arrow[ur, two heads] \arrow[rr, two heads] & & \text{uOBDD} \arrow[uu] \arrow[ur, two heads] \arrow[rr, two heads] & & \text{nOBDD} \arrow[uu] \arrow[ur, two heads] &
\end{tikzcd}$$



The intensional approach for self-join-free CQs

Theorem [Olteanu and Huang, 2008]

For any hierarchical self-join-free CQ Q, given a TID $\mathbf{I}$,

The intensional approach for self-join-free CQs

Theorem [Olteanu and Huang, 2008]

For any hierarchical self-join-free CQ Q, given a TID I, we can compute in linear time a read-once formula representing the Boolean provenance of Q on I

The intensional approach for self-join-free CQs

Theorem [Olteanu and Huang, 2008]

For any hierarchical self-join-free CQ Q, given a TID I, we can compute in linear time a read-once formula representing the Boolean provenance of Q on I

Proof: just follow the previous algorithm and its independent ANDs and ORs

The intensional approach for self-join-free CQs

Theorem [Olteanu and Huang, 2008]

For any hierarchical self-join-free CQ Q, given a TID I, we can compute in linear time a read-once formula representing the Boolean provenance of Q on I

Proof: just follow the previous algorithm and its independent ANDs and ORs

Corollary

For any hierarchical self-join-free CQ Q, the problem $\mathrm{PQE}(Q)$ is in linear time up to the cost of arithmetic operations

Other results for the intensional approach

- For UCQs, results in [Jha and Suciu, 2013]:
    - Characterization of the queries for which we can compute **read-once provenance**

Other results for the intensional approach

- For UCQs, results in [Jha and Suciu, 2013]:
    - Characterization of the queries for which we can compute read-once provenance
    - Characterization of the queries for which we can compute OBDD provenance

Other results for the intensional approach

- For UCQs, results in [Jha and Suciu, 2013]:
  - Characterization of the queries for which we can compute **read-once provenance**
  - Characterization of the queries for which we can compute **OBDD provenance**
  - Sufficient conditions to have **FBDDs** and **d-DNNFs**

Other results for the intensional approach

- For UCQs, results in [Jha and Suciu, 2013]:
    - Characterization of the queries for which we can compute **read-once provenance**
    - Characterization of the queries for which we can compute **OBDD provenance**
    - Sufficient conditions to have **FBDDs** and **d-DNNFs**

- For some safe UCQs we cannot compute provenance as **DLDDs** [Beame et al., 2017]

Other results for the intensional approach

- For UCQs, results in [Jha and Suciu, 2013]:
    - Characterization of the queries for which we can compute **read-once provenance**
    - Characterization of the queries for which we can compute **OBDD provenance**
    - Sufficient conditions to have **FBDDs** and **d-DNNFs**
- For some safe UCQs we cannot compute provenance as **DLDDs** [Beame et al., 2017]
- For some safe UCQs we cannot compute **d-SDNNFs** [Bova and Szeider, 2017]

Other results for the intensional approach

- For UCQs, results in [Jha and Suciu, 2013]:
  - Characterization of the queries for which we can compute **read-once provenance**
  - Characterization of the queries for which we can compute **OBDD provenance**
  - Sufficient conditions to have **FBDDs** and **d-DNNFs**
- For some safe UCQs we cannot compute provenance as **DLDDs** [Beame et al., 2017]
- For some safe UCQs we cannot compute **d-SDNNFs** [Bova and Szeider, 2017]
- Good candidate: **d-DNNF**, or **d-D** (allows arbitrary negations)

  → Note: it's **open** whether d-DNNFs and d-Ds are indeed different :)

Other results for the intensional approach

- For UCQs, results in [Jha and Suciu, 2013]:
    - Characterization of the queries for which we can compute **read-once provenance**
    - Characterization of the queries for which we can compute **OBDD provenance**
    - Sufficient conditions to have **FBDDs** and **d-DNNFs**
- For some safe UCQs we cannot compute provenance as **DLDDs** [Beame et al., 2017]
- For some safe UCQs we cannot compute **d-SDNNFs** [Bova and Szeider, 2017]
- Good candidate: **d-DNNF**, or **d-D** (allows arbitrary negations)

  → Note: it's **open** whether d-DNNFs and d-Ds are indeed different :)
- **Crux of the problem:** capture arithmetic operations on probabilities with a d-D circuit, specifically **inclusion-exclusion**; see [Monet, 2020]

References i

Amarilli, A., Capelli, F., Monet, M., and Senellart, P. (2019).
Connecting knowledge compilation classes and width parameters.
In *ToCS*, number 2019.

Beame, P., Li, J., Roy, S., and Suciu, D. (2017).
Exact model counting of query expressions: Limitations of propositional methods.
TODS, 42(1):1.

Bova, S. and Szeider, S. (2017).
Circuit treewidth, sentential decision, and query compilation.
In *PODS*. ACM.

References ii

- Deutch, D., Milo, T., Roy, S., and Tannen, V. (2014).
  Circuits for Datalog provenance.
  In *ICDT*.
- Green, T. J., Karvounarakis, G., and Tannen, V. (2007).
  Provenance semirings.
  In *PODS*.
- Jha, A. and Suciu, D. (2013).
  Knowledge compilation meets database theory: Compiling queries to decision diagrams.
  Theory of Computing Systems, 52(3).

References iii

Monet, M. (2020).
Solving a special case of the intensional vs extensional conjecture in probabilistic databases.
In *PODS*.

Olteanu, D. and Huang, J. (2008).
Using OBDDs for efficient query evaluation on probabilistic databases.
In *SUM*. Springer.

Ordered Binary Decision Diagram (OBDD)

OBDD for a Boolean query Q on database I:
ordered decision diagram on the facts of I to decide whether Q holds

Ordered Binary Decision Diagram (OBDD)

OBDD for a Boolean query Q on database I:
ordered decision diagram on the facts of I to decide whether Q holds

$$Q : \pi_{\emptyset}(R \bowtie S \bowtie T)$$

R	
a	r_1
b	r_2
c	r_3

S		
a	a	s_1
b	v	s_2
b	w	s_3

T	
v	t_1
w	t_2
b	t_3

Ordered Binary Decision Diagram (OBDD)

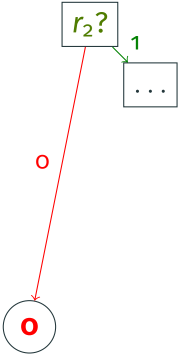
OBDD for a Boolean query Q on database I:
ordered decision diagram on the facts of I to decide whether Q holds

$$Q : \pi_{\emptyset}(R \bowtie S \bowtie T)$$

R	
a	r_1
b	r_2
c	r_3

S		
a	a	s_1
b	v	s_2
b	w	s_3

T	
v	t_1
w	t_2
b	t_3



Ordered Binary Decision Diagram (OBDD)

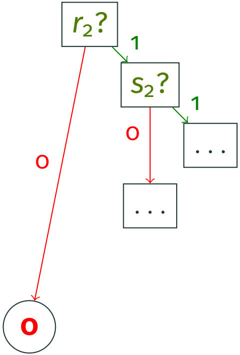
OBDD for a Boolean query Q on database I:
ordered decision diagram on the facts of I to decide whether Q holds

$$Q : \pi_{\emptyset}(R \bowtie S \bowtie T)$$

R	
a	r_1
b	r_2
c	r_3

S		
a	a	s_1
b	v	s_2
b	w	s_3

T	
v	t_1
w	t_2
b	t_3



Ordered Binary Decision Diagram (OBDD)

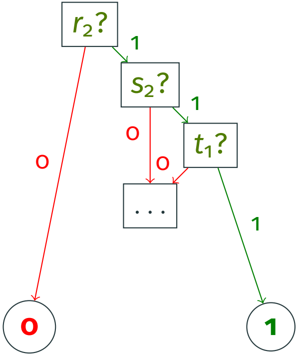
OBDD for a Boolean query Q on database I:
ordered decision diagram on the facts of I to decide whether Q holds

$$Q : \pi_{\emptyset}(R \bowtie S \bowtie T)$$

R	
a	r_1
b	r_2
c	r_3

S		
a	a	s_1
b	v	s_2
b	w	s_3

T	
v	t_1
w	t_2
b	t_3



Ordered Binary Decision Diagram (OBDD)

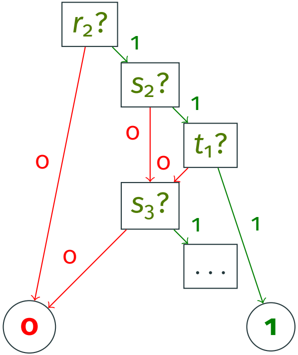
OBDD for a Boolean query Q on database I:
ordered decision diagram on the facts of I to decide whether Q holds

$$Q : \pi_{\emptyset}(R \bowtie S \bowtie T)$$

R	
a	r_1
b	r_2
c	r_3

S		
a	a	s_1
b	v	s_2
b	w	s_3

T	
v	t_1
w	t_2
b	t_3



Ordered Binary Decision Diagram (OBDD)

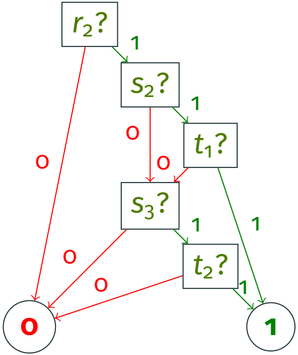
OBDD for a Boolean query Q on database I:
ordered decision diagram on the facts of I to decide whether Q holds

$$Q : \pi_{\emptyset}(R \bowtie S \bowtie T)$$

R	
a	r_1
b	r_2
c	r_3

S		
a	a	s_1
b	v	s_2
b	w	s_3

T	
v	t_1
w	t_2
b	t_3



Ordered Binary Decision Diagram (OBDD)

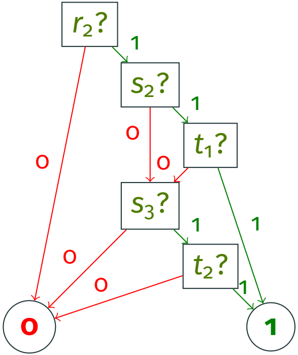
OBDD for a Boolean query Q on database I:
ordered decision diagram on the facts of I to decide whether Q holds

$$Q : \pi_{\emptyset}(R \bowtie S \bowtie T)$$

R	
a	r_1
b	r_2
c	r_3

S		
a	a	s_1
b	v	s_2
b	w	s_3

T	
v	t_1
w	t_2
b	t_3



→ We can compute the probability of an OBDD **bottom-up**

Probabilistic Databases: Width-Based Approaches

EDBT-Intended Summer School

Antoine Amarilli

Going back to more restricted instances

OK, PQE *is* ***intractable*** *for essentially all queries. What now?*

Going back to more restricted instances

OK, PQE *is* ***intractable*** *for essentially all queries. What now?*

- We could restrict the **structure** of instances: instead of arbitrary graphs, focus on:
    - probabilistic **words**
    - probabilistic **trees**
    - probabilistic graphs with **bounded treewidth**

Going back to more restricted instances

*OK, PQE is **intractable** for essentially all queries. What now?*

- We could restrict the **structure** of instances: instead of arbitrary graphs, focus on:
  - probabilistic **words**
  - probabilistic **trees**
  - probabilistic graphs with **bounded treewidth**
- In the non-probabilistic case, this ensures tractability for **complex queries**

→ Could the same be true in the **probabilistic case**?

Going back to more restricted instances

*OK, PQE is **intractable** for essentially all queries. What now?*

- We could restrict the **structure** of instances: instead of arbitrary graphs, focus on:
  - probabilistic **words**
  - probabilistic **trees**
  - probabilistic graphs with **bounded treewidth**
- In the non-probabilistic case, this ensures tractability for **complex queries**

→ Could the same be true in the **probabilistic case**?

Theorem (A., Bourhis, Senellart, 2015, 2016)

*Fix a bound $k \in \mathbb{N}$ and fix a Boolean **monadic second-order** query Q. Then $\mathrm{PQE}(Q)$ is in **PTIME** on input TID instances of **treewidth** $\leq k$*

Going back to more restricted instances

OK, PQE *is **intractable** for essentially all queries. What now?*

- We could restrict the **structure** of instances: instead of arbitrary graphs, focus on:
  - probabilistic **words**
  - probabilistic **trees**
  - probabilistic graphs with **bounded treewidth**
- In the non-probabilistic case, this ensures tractability for **complex queries**

→ Could the same be true in the **probabilistic case**?

Theorem (A., Bourhis, Senellart, 2015, 2016)

Fix a bound $k \in \mathbb{N}$ *and fix a Boolean **monadic second-order** query* Q*. Then* $\mathrm{PQE}(Q)$ *is in **PTIME** on input TID instances of **treewidth*** $\leq k$

Conversely, there is a query Q *for which* $\mathrm{PQE}(Q)$ *is intractable on **any** input instance family of unbounded treewidth (under some technical assumptions)*

Non-probabilistic query evaluation on words

Database: a **word** w where nodes have a color from an alphabet ○ 🔴 🔵

Non-probabilistic query evaluation on words

Database: a **word** w where nodes have a color from an alphabet ○ 🔴 🔵

Query Q: a **sentence** (yes/no question) in **monadic second-order logic** (MSO)

“Is there both a pink and a blue node?”

Non-probabilistic query evaluation on words

Database: a **word** w where nodes have a color from an alphabet ○ 🔴 🔵

Query Q: a **sentence** (yes/no question) in **monadic second-order logic** (MSO)

"Is there both a pink and a blue node?"

Result: TRUE/FALSE indicating if the word w satisfies the query Q

Non-probabilistic query evaluation on words

Database: a **word** w where nodes have a color from an alphabet ○ 🔴 🔵

Query Q: a **sentence** (yes/no question) in **monadic second-order logic** (MSO)

“Is there both a pink and a blue node?”

Result: TRUE/FALSE indicating if the word w satisfies the query Q

Computational complexity as a function of w
(the query Q is **fixed**)

Monadic second-order logic (MSO)

- $P_{\text{🔵}}(x)$ means “x is blue”; also $P_{\text{🔴}}(x)$, $P_{\bigcirc}(x)$
- $x \rightarrow y$ means “x is the predecessor of y”

Monadic second-order logic (MSO)

- $P_{🔵}(x)$ means "x is blue"; also $P_{🔴}(x)$, $P_{○}(x)$
- $x \to y$ means "x is the predecessor of y"

- **Propositional logic:** formulas with **AND** $\wedge$, **OR** $\vee$, **NOT** $\neg$
  - $P_{🔴}(x) \wedge P_{🔵}(y)$ means *"Node **x** is pink and node **y** is blue"*

Monadic second-order logic (MSO)

- $P_{\text{🔵}}(x)$ means "x is blue"; also $P_{\text{🔴}}(x)$, $P_{\bigcirc}(x)$
- $x \rightarrow y$ means "x is the predecessor of y"

- **Propositional logic:** formulas with **AND** $\wedge$, **OR** $\vee$, **NOT** $\neg$
  - $P_{\text{🔴}}(x) \wedge P_{\text{🔵}}(y)$ means *"Node x is pink and node y is blue"*

- **First-order logic:** adds **existential quantifier** $\exists$ and **universal quantifier** $\forall$
  - $\exists x\, y\; P_{\text{🔴}}(x) \wedge P_{\text{🔵}}(y)$ means *"There is both a pink and a blue node"*

Monadic second-order logic (MSO)

- $P_{🔵}(x)$ means "x is blue"; also $P_{🔴}(x)$, $P_{○}(x)$
- $x \rightarrow y$ means "x is the predecessor of y"

- **Propositional logic:** formulas with **AND** $\wedge$, **OR** $\vee$, **NOT** $\neg$
  - $P_{🔴}(x) \wedge P_{🔵}(y)$ means *"Node x is pink and node y is blue"*

- **First-order logic:** adds **existential quantifier** $\exists$ and **universal quantifier** $\forall$
  - $\exists x\, y\; P_{🔴}(x) \wedge P_{🔵}(y)$ means *"There is both a pink and a blue node"*

- **Monadic second-order logic (MSO):** adds **quantifiers over sets**
  - $\exists S\; \forall x\; S(x)$ means *"there is a set S containing every element x"*
  - Can express **transitive closure** $x \rightarrow^{*} y$, i.e., *"x is before y"*
  - $\forall x\, P_{🔴}(x) \Rightarrow \exists y\, P_{🔵}(y) \wedge x \rightarrow^{*} y$
    means *"There is a blue node after every pink node"*

Word automata

Translate the query Q to a deterministic word automaton

Alphabet: ○ 🔴 🔵

w: ○—🔴—○—🔵—○

Q: $\exists x\, y\ P_{\text{🔴}}(x) \land P_{\text{🔵}}(y)$

Word automata

Translate the query Q to a **deterministic word automaton**

Alphabet: ⚪ 🔴 🔵   **w:** ⚪—🔴—⚪—🔵—⚪   **Q:** $\exists x\, y\ P_{\text{🔴}}(x) \wedge P_{\text{🔵}}(y)$

- **States:** $\{\bot, B, P, \top\}$

Word automata

Translate the query ***Q*** to a **deterministic word automaton**

Alphabet: ○ 🔴 🔵

***w*:**

***Q*:** $\exists x\, y\; P_{🔴}(x) \wedge P_{🔵}(y)$

- **States:** $\{\bot, B, P, \top\}$
- **Final states:** $\{\top\}$

Word automata

Translate the query Q to a **deterministic word automaton**

Alphabet: ○ 🔴 🔵

w:

Q: $\exists x\, y\; P_{🔴}(x) \land P_{🔵}(y)$

- **States:** $\{\bot, B, P, \top\}$
- **Final states:** $\{\top\}$
- **Initial function:** ○ $\bot$ 🔴 P 🔵 B

Word automata

Translate the query **Q** to a **deterministic word automaton**

Alphabet: ○ 🔴 🔵

w:

Q: $\exists x\, y\; P_{\text{🔴}}(x) \wedge P_{\text{🔵}}(y)$

- **States:** $\{\bot, B, P, \top\}$
- **Final states:** $\{\top\}$
- **Initial function:** ○ $\bot$ 🔴 P 🔵 B

Word automata

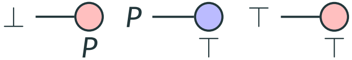
Translate the query Q to a **deterministic word automaton**

Alphabet: ○ 🔴 🔵

w:

Q: $\exists x\, y\; P_{🔴}(x) \land P_{🔵}(y)$

- **States:** $\{\bot, B, P, \top\}$
- **Final states:** $\{\top\}$
- **Initial function:** ○ $\bot$ 🔴 P 🔵 B
- **Transitions** (examples):



Word automata

Translate the query Q to a **deterministic word automaton**

Alphabet: ○ 🔴 🔵

w:

Q: $\exists x\, y\; P_{🔴}(x) \wedge P_{🔵}(y)$

- **States:** $\{\bot, B, P, \top\}$
- **Final states:** $\{\top\}$
- **Initial function:** ○ $\bot$ 🔴 P 🔵 B
- **Transitions** (examples):

Word automata

Translate the query Q to a **deterministic word automaton**

Alphabet: ○ 🔴 🔵

w:

Q: $\exists x\, y\; P$🔴$(x) \wedge P$🔵(y)

- **States:** $\{\bot, B, P, \top\}$
- **Final states:** $\{\top\}$
- **Initial function:** ○ $\bot$ 🔴 P 🔵 B
- **Transitions** (examples):

Word automata

Translate the query Q to a **deterministic word automaton**

Alphabet: ○ 🔴 🔵

w:

Q: $\exists x\, y\; P_{\text{🔴}}(x) \land P_{\text{🔵}}(y)$

- **States:** $\{\perp, B, P, \top\}$
- **Final states:** $\{\top\}$
- **Initial function:** ○ $\perp$ 🔴 P 🔵 B
- **Transitions** (examples):

Word automata

Translate the query Q to a **deterministic word automaton**

Alphabet: ○ 🔴 🔵

w:

Q: $\exists x\, y\; P_{🔴}(x) \wedge P_{🔵}(y)$

- **States:** $\{\perp, B, P, \top\}$
- **Final states:** $\{\top\}$
- **Initial function:** ○ $\perp$ 🔴 P 🔵 B
- **Transitions** (examples): $\perp \xrightarrow{🔴} P$ $P \xrightarrow{🔵} \top$ $\top \xrightarrow{🔴} \top$

Word automata

Translate the query Q to a deterministic word automaton

Alphabet: ○ 🔴 🔵

w:

Q: $\exists x\, y\; P_{\text{🔴}}(x) \wedge P_{\text{🔵}}(y)$

- **States:** $\{\bot, B, P, \top\}$
- **Final states:** $\{\top\}$
- **Initial function:** ○ $\bot$ 🔴 P 🔵 B
- **Transitions** (examples):

Theorem (Büchi, 1960)

MSO and word automata have the same expressive power on words

Word automata

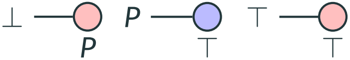
Translate the query **Q** to a deterministic word automaton

Alphabet: ○ 🔴 🔵

w:

Q: $\exists x\, y\;$ P🔴$(x) \wedge P$🔵(y)

- States: $\{\bot, B, P, \top\}$
- Final states: $\{\top\}$
- Initial function: ○ $\bot$ 🔴 P 🔵 B
- Transitions (examples):



Theorem (Büchi, 1960)

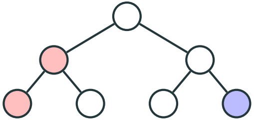
MSO and word automata have the same expressive power on words

Corollary

Query evaluation of MSO on words is in linear time (in data complexity)

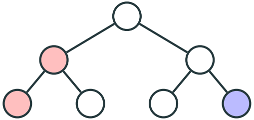
Non-probabilistic query evaluation on trees

Database: a **tree** T where nodes have a color from an alphabet ○ 🟥 🟦



Non-probabilistic query evaluation on trees

Database: a **tree** T where nodes have a color from an alphabet ○ 🔴 🔵



Query Q: in monadic second-order logic (MSO)

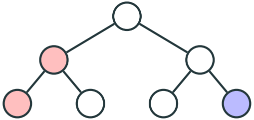
- $P_{🔵}(x)$ means "x is blue"
- $x \rightarrow y$ means "x is the parent of y"

"Is there both a pink and a blue node?"

$\exists x\, y\; P_{🔴}(x) \wedge P_{🔵}(y)$

Non-probabilistic query evaluation on trees

Database: a **tree** T where nodes have a color from an alphabet ○ 🔴 🔵



Query Q: in monadic second-order logic (MSO)

- $P_{🔵}(x)$ means "x is blue"
- $x \rightarrow y$ means "x is the parent of y"

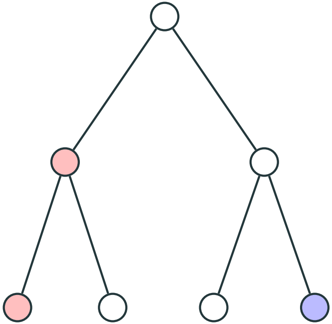
"Is there both a pink and a blue node?"

$\exists x\, y\ P_{🔴}(x) \wedge P_{🔵}(y)$

Result: YES/NO indicating if the tree T satisfies the query Q

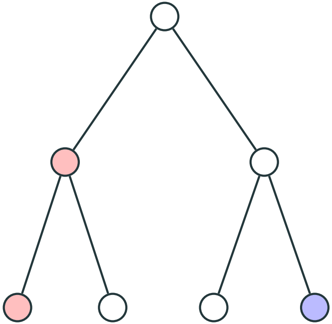
Tree automata

Tree alphabet: ○ 🔴 🔵



Tree automata

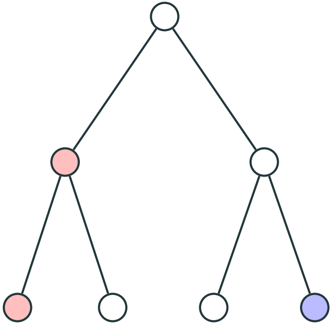
Tree alphabet: ○ 🔴 🔵



- Bottom-up deterministic **tree automaton**
- *“Is there both a pink and a blue node?”*

Tree automata

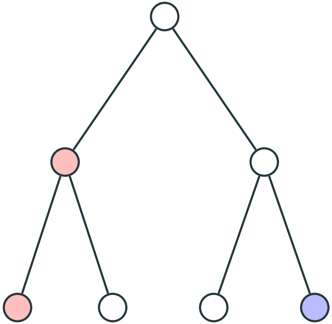
Tree alphabet: ○ 🔴 🔵



- Bottom-up deterministic **tree automaton**
- *“Is there both a pink and a blue node?”*
- **States:** $\{\perp, B, P, \top\}$

Tree automata

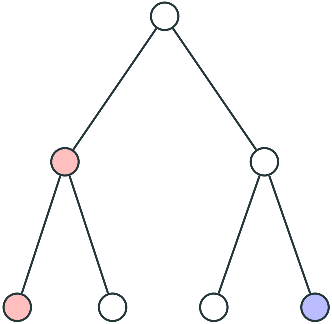
Tree alphabet: ○ 🔴 🔵



- Bottom-up deterministic **tree automaton**
- *“Is there both a pink and a blue node?”*
- **States:** $\{\bot, B, P, \top\}$
- **Final states:** $\{\top\}$

Tree automata

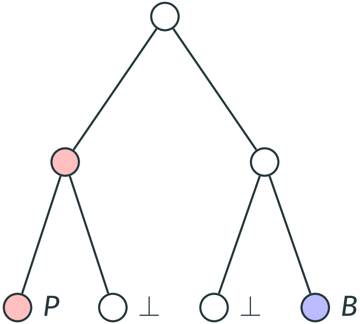
Tree alphabet: ○ 🔴 🔵



- Bottom-up deterministic **tree automaton**
- *"Is there both a pink and a blue node?"*
- States: $\{\bot, B, P, \top\}$
- Final states: $\{\top\}$
- Initial function: ○ $\bot$ 🔴 P 🔵 B

Tree automata

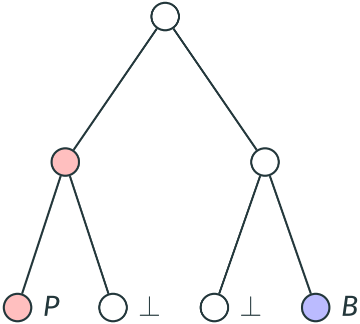
Tree alphabet: ○ 🔴 🔵



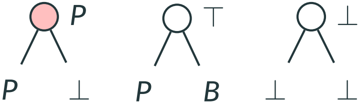
- Bottom-up deterministic **tree automaton**
- *"Is there both a pink and a blue node?"*
- **States:** $\{\bot, B, P, \top\}$
- **Final states:** $\{\top\}$
- **Initial function:** ○ $\bot$ 🔴 P 🔵 B

Tree automata

Tree alphabet: ○ 🔴 🔵

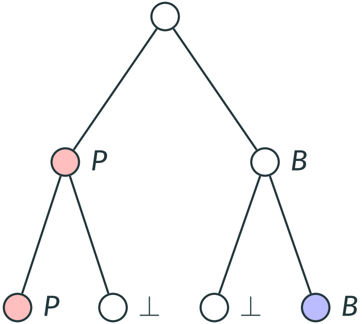


- Bottom-up deterministic **tree automaton**
- *"Is there both a pink and a blue node?"*
- **States:** $\{\bot, B, P, \top\}$
- **Final states:** $\{\top\}$
- **Initial function:** ○ $\bot$ 🔴 P 🔵 B
- **Transitions** (examples):

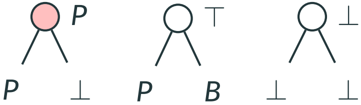


Tree automata

Tree alphabet: ○ 🔴 🔵

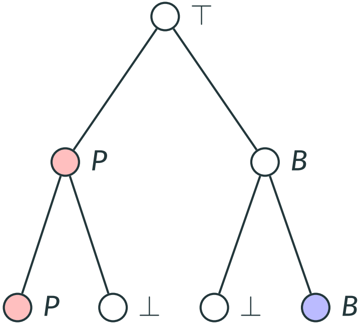


- Bottom-up deterministic **tree automaton**
- *"Is there both a pink and a blue node?"*
- **States:** $\{\bot, B, P, \top\}$
- **Final states:** $\{\top\}$
- **Initial function:** ○ $\bot$ 🔴 P 🔵 B
- **Transitions** (examples):

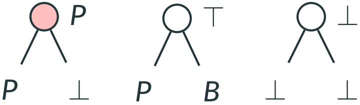


Tree automata

Tree alphabet: ○ 🔴 🔵

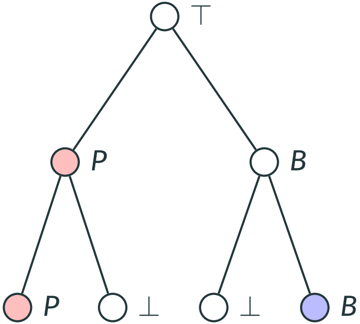


- Bottom-up deterministic **tree automaton**
- *“Is there both a pink and a blue node?”*
- **States:** $\{\bot, B, P, \top\}$
- **Final states:** $\{\top\}$
- **Initial function:** ○ $\bot$ 🔴 P 🔵 B
- **Transitions** (examples):

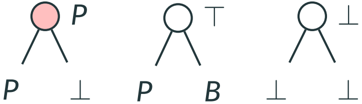


Tree automata

Tree alphabet: ○ 🔴 🔵



- Bottom-up deterministic tree automaton
- *"Is there both a pink and a blue node?"*
- States: $\{\bot, B, P, \top\}$
- Final states: $\{\top\}$
- Initial function: ○ $\bot$ 🔴 P 🔵 B
- Transitions (examples):



Theorem ([Thatcher and Wright, 1968])

MSO and tree automata have the same expressive power on trees

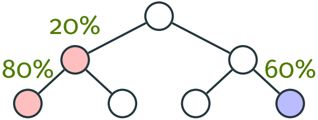
Probabilistic query evaluation on trees

Let's now define the PQE problem for MSO queries on trees:

Probabilistic query evaluation on trees

Let's now define the PQE problem for MSO queries on trees:

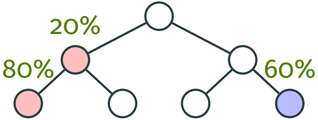
Database: a tree T where each node has a probability of keeping its color (vs taking the default color ○)



Probabilistic query evaluation on trees

Let’s now define the **PQE problem** for MSO queries on trees:

Database: a **tree** T where each node has a probability of **keeping its color** (vs taking the **default color** ○)



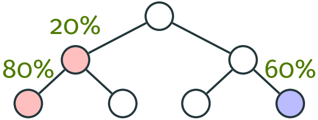
Query Q: in monadic second-order logic (MSO)

$\exists x\, y\; P_{🔴}(x) \wedge P_{🔵}(y)$

Probabilistic query evaluation on trees

Let’s now define the **PQE problem** for MSO queries on trees:

Database: a **tree** T where each node has a probability of **keeping its color** (vs taking the **default color** ○)



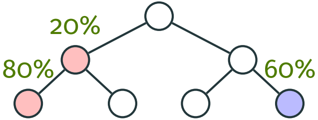
Query Q: in monadic second-order logic (MSO)

$\exists x\, y\; P_{●}(x) \land P_{●}(y)$

Result: **probability** that the probabilistic tree T satisfies the query Q

Probabilistic query evaluation on trees

Let's now define the **PQE problem** for MSO queries on trees:



Database: a **tree** T where each node has a probability of **keeping its color** (vs taking the **default color** ○)

Query Q: in monadic second-order logic (MSO)

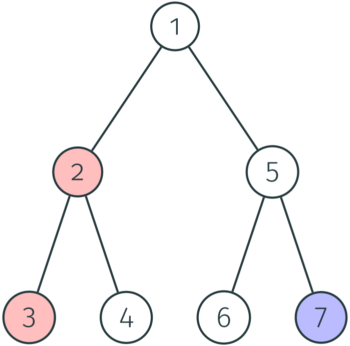
$\exists x\, y\, P_{\bullet}(x) \wedge P_{\bullet}(y)$

Result: **probability** that the probabilistic tree T satisfies the query Q

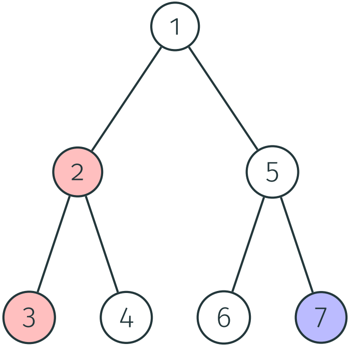
Theorem

*For any fixed **MSO query** Q, the problem $\mathrm{PQE}(Q)$ on trees is in **linear time** assuming constant-time arithmetics*

Uncertain trees: capturing how the query result depends on the choices

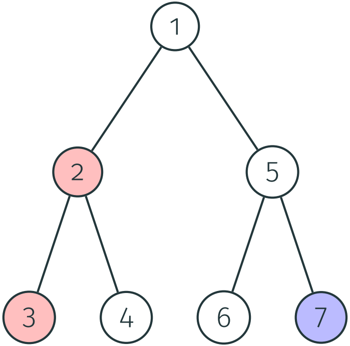


Uncertain trees: capturing how the query result depends on the choices



A **valuation** of a tree decides whether to **keep** (1) or **discard** (0) node labels

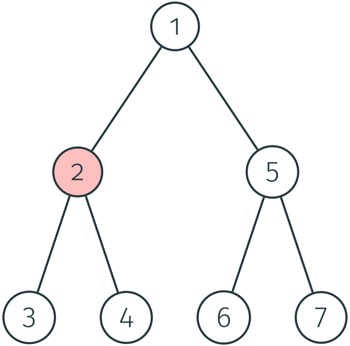
Uncertain trees: capturing how the query result depends on the choices



A **valuation** of a tree decides whether to **keep** (1) or **discard** (0) node labels

Valuation: $\{2, 3, 7 \mapsto 1, \quad * \mapsto 0\}$

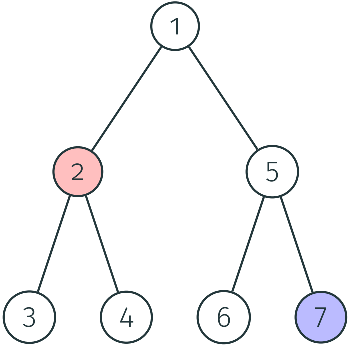
Uncertain trees: capturing how the query result depends on the choices



A **valuation** of a tree decides whether to **keep** (1) or **discard** (0) node labels

Valuation: $\{\mathbf{2} \mapsto \mathbf{1},\ \ast \mapsto \mathbf{0}\}$

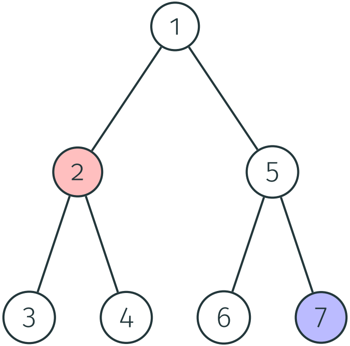
Uncertain trees: capturing how the query result depends on the choices



A **valuation** of a tree decides whether to **keep** (1) or **discard** (0) node labels

Valuation: $\{2, 7 \mapsto 1, \quad * \mapsto 0\}$

Uncertain trees: capturing how the query result depends on the choices

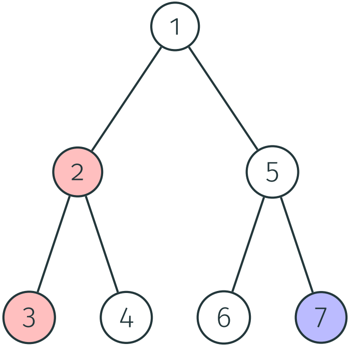


A **valuation** of a tree decides whether to **keep** (1) or **discard** (0) node labels

Valuation: $\{2, 7 \mapsto 1, \quad * \mapsto 0\}$

Q: *"Is there both a pink and a blue node?"*

Uncertain trees: capturing how the query result depends on the choices



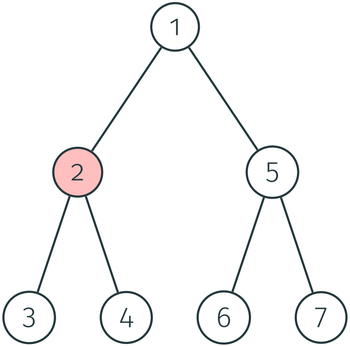
A **valuation** of a tree decides whether to **keep** (1) or **discard** (0) node labels

Valuation: $\{2, 3, 7 \mapsto 1, \quad * \mapsto 0\}$

Q: *"Is there both a pink and a blue node?"*

The query *Q* returns YES

Uncertain trees: capturing how the query result depends on the choices



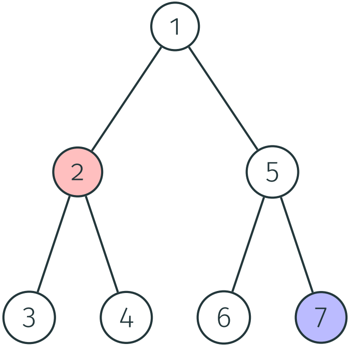
A **valuation** of a tree decides whether to **keep** (1) or **discard** (0) node labels

Valuation: $\{2 \mapsto 1, \ * \mapsto 0\}$

Q: *“Is there both a pink and a blue node?”*

The query *Q* returns **NO**

Uncertain trees: capturing how the query result depends on the choices



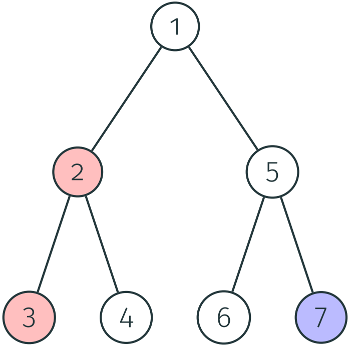
A **valuation** of a tree decides whether to **keep** (1) or **discard** (0) node labels

Valuation: $\{2, 7 \mapsto 1, \ * \mapsto 0\}$

Q: *"Is there both a pink and a blue node?"*

The query *Q* returns YES

Uncertain trees: capturing how the query result depends on the choices

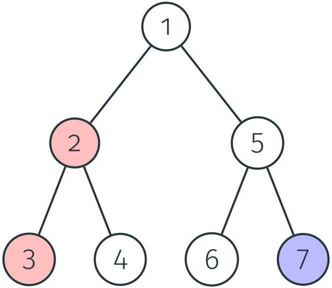


A **valuation** of a tree decides whether to **keep** (1) or **discard** (0) node labels

Q: *“Is there both a pink and a blue node?”*

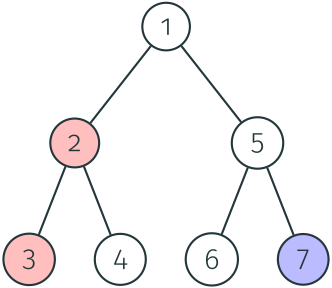
→ This is just a **Boolean provenance circuit** on the “color facts” of the tree nodes!

Example: Provenance circuit



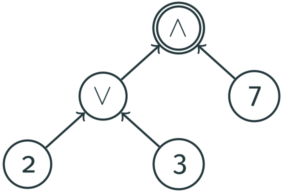
Query: *Is there both a pink and a blue node?*

Example: Provenance circuit

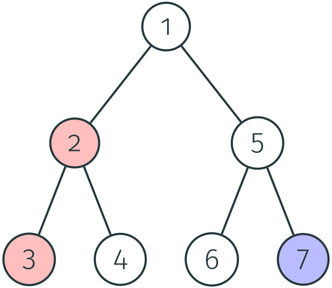


Query: *Is there both a pink and a blue node?*

Provenance circuit:

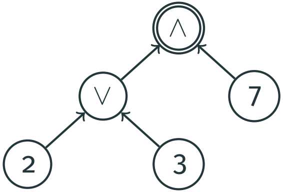


Example: Provenance circuit



Query: *Is there both a pink and a blue node?*

Provenance circuit:



Formal definition of provenance circuits:

- Boolean query Q, uncertain tree T, circuit C
- **Variable gates** of C: nodes of T
- **Condition:** Let ν be a valuation of T, then $\nu(C)$ iff $\nu(T)$ satisfies Q

Provenance circuits on trees

Theorem

For any bottom-up tree automaton A and input tree T, we can build a Boolean SDNNF provenance circuit of A on T in $O(|A| \times |T|)$

Provenance circuits on trees

Theorem

For any bottom-up tree automaton A and input tree T,
we can build a Boolean SDNNF provenance circuit of A on T in $O(|A| \times |T|)$

- **Alphabet:** ○ 🔴 🔵
- **Automaton:** *“Is there both a pink and a blue node?”*

- **States:** $\{\bot, B, P, \top\}$
- **Final:** $\{\top\}$

- **Transitions:**

Provenance circuits on trees

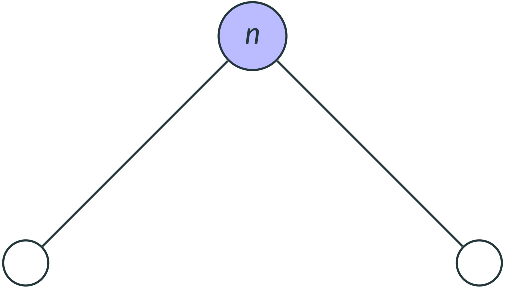
Theorem

For any bottom-up tree automaton A and input tree T, we can build a Boolean SDNNF provenance circuit of A on T in $O(|A| \times |T|)$

- **Alphabet:** ○ 🔴 🔵
- **Automaton:** *“Is there both a pink and a blue node?”*

- **States:** $\{\bot, B, P, \top\}$
- **Final:** $\{\top\}$

- **Transitions:**



Provenance circuits on trees

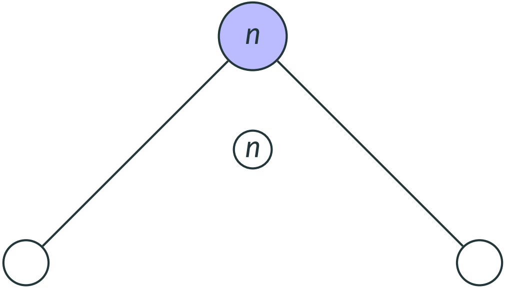
Theorem

For any bottom-up ***tree automaton*** *A and input* ***tree*** *T, we can build a Boolean* ***SDNNF provenance circuit*** *of A on T in $O(|A| \times |T|)$*

- **Alphabet:** 
- **Automaton:** *“Is there both a pink and a blue node?”*

- **States:** $\{\bot, B, P, \top\}$
- **Final:** $\{\top\}$

- **Transitions:**



Provenance circuits on trees

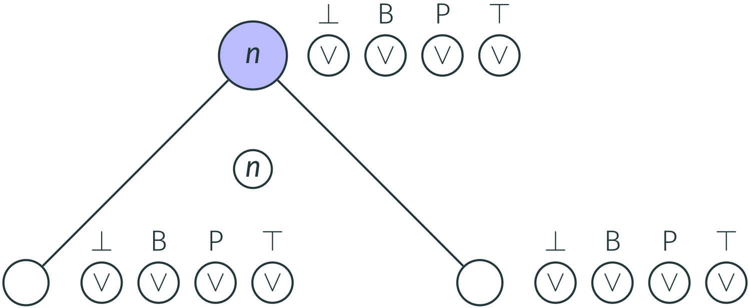
Theorem

For any bottom-up tree automaton A and input tree T, we can build a Boolean SDNNF provenance circuit of A on T in $O(|A| \times |T|)$

- **Alphabet:** ○ 🔴 🔵
- **Automaton:** *"Is there both a pink and a blue node?"*

- **States:** $\{\bot, B, P, \top\}$
- **Final:** $\{\top\}$

- **Transitions:**

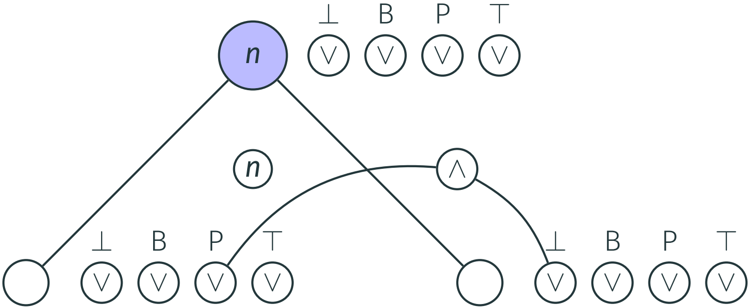


Provenance circuits on trees

Theorem

For any bottom-up *tree automaton* A *and input* *tree* T,
we can build a Boolean *SDNNF provenance circuit* *of* A *on* T *in* $O(|A| \times |T|)$

- **Alphabet:** ○ 🔴 🔵
- **Automaton:** *“Is there both a pink and a blue node?”*
- **States:** $\{\bot, B, P, \top\}$
- **Final:** $\{\top\}$
- **Transitions:**

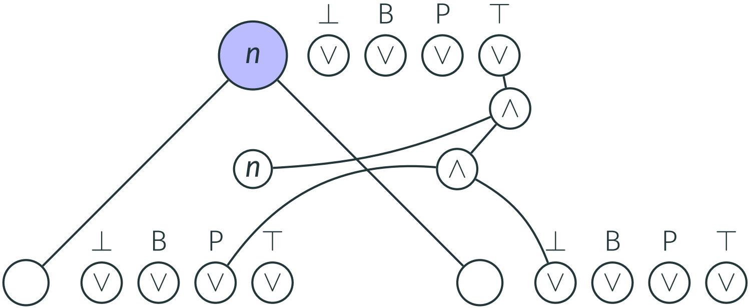


Provenance circuits on trees

Theorem

For any bottom-up *tree automaton* A *and input* *tree* T*,*
we can build a Boolean *SDNNF provenance circuit* *of* A *on* T *in* $O(|A| \times |T|)$

- **Alphabet:** ○ 🔴 🔵
- **Automaton:** *“Is there both a pink and a blue node?”*
- **States:** $\{\bot, B, P, \top\}$
- **Final:** $\{\top\}$
- **Transitions:**



Provenance circuits on trees

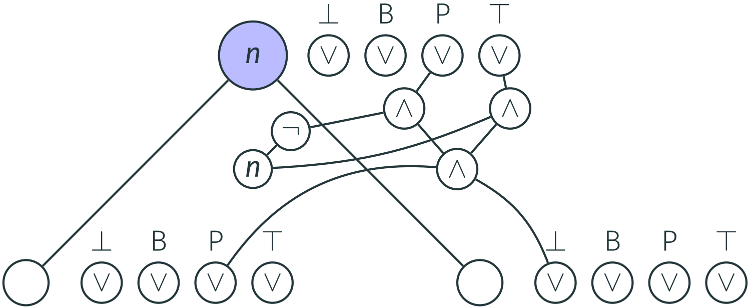
Theorem

For any bottom-up tree automaton A and input tree T, we can build a Boolean SDNNF provenance circuit of A on T in $O(|A| \times |T|)$

- **Alphabet:** ○ 🔴 🔵
- **Automaton:** *“Is there both a pink and a blue node?”*

- **States:** $\{\bot, B, P, \top\}$
- **Final:** $\{\top\}$

- **Transitions:**



Provenance circuits on trees

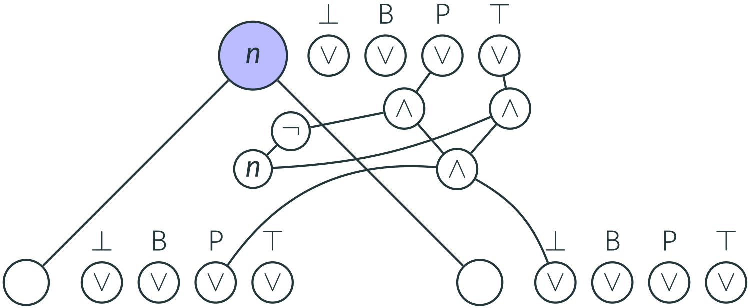
Theorem

For any bottom-up ***unambiguous tree automaton*** A *and input* ***tree*** T,
we can build a Boolean ***d-SDNNF provenance circuit*** *of* A *on* T *in* $O(|A| \times |T|)$

- **Alphabet:** ○ 🔴 🔵
- **Automaton:** *"Is there both a pink and a blue node?"*

- **States:** $\{\bot, B, P, \top\}$
- **Final:** $\{\top\}$

- **Transitions:**



Connections to knowledge compilation

The provenance circuits of automata on trees are…

Connections to knowledge compilation

The provenance circuits of automata on trees are…

- DNNF circuits:
  - → Negations only at the leaves
  - → Conjunctions are between disjoint subtrees

Connections to knowledge compilation

The provenance circuits of automata on trees are...

- **DNNF** circuits:
  - → Negations only at the **leaves**
  - → Conjunctions are between **disjoint** subtrees
- **Structured** circuits
  - → The v-tree follows the **shape** of the input tree

Connections to knowledge compilation

The provenance circuits of automata on trees are...

- **DNNF** circuits:
  - → Negations only at the **leaves**
  - → Conjunctions are between **disjoint** subtrees
- **Structured** circuits
  - → The v-tree follows the **shape** of the input tree
- **d-SDNNFs** when the input automaton is **unambiguous**

Connections to knowledge compilation

The provenance circuits of automata on trees are...

- DNNF circuits:
  - → Negations only at the leaves
  - → Conjunctions are between disjoint subtrees
- Structured circuits
  - → The v-tree follows the shape of the input tree
- d-SDNNFs when the input automaton is unambiguous
- Of width bounded by the number of states of the automaton [Capelli and Mengel, 2019]

Connections to knowledge compilation

The provenance circuits of automata on trees are...

- DNNF circuits:
  - → Negations only at the leaves
  - → Conjunctions are between disjoint subtrees
- Structured circuits
  - → The v-tree follows the shape of the input tree
- d-SDNNFs when the input automaton is unambiguous
- Of width bounded by the number of states of the automaton [Capelli and Mengel, 2019]

→ Remark: for words, we obtain diagrams (OBDDs, etc.)

Connections to knowledge compilation

The provenance circuits of automata on trees are...

- **DNNF** circuits:
  - → Negations only at the **leaves**
  - → Conjunctions are between **disjoint** subtrees
- **Structured** circuits
  - → The v-tree follows the **shape** of the input tree
- **d-SDNNFs** when the input automaton is **unambiguous**
- Of **width** bounded by the number of **states** of the automaton [Capelli and Mengel, 2019]

→ Remark: for **words**, we obtain **diagrams** (OBDDs, etc.)

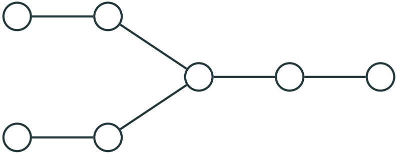
Corollary

For any MSO query Q*, the problem* $\mathrm{PQE}(Q)$ *on probabilistic trees is in* ***linear time*** *assuming constant-time arithmetics*

Treewidth

We have shown tractability of PQE on **trees**; let us extend to **bounded treewidth**

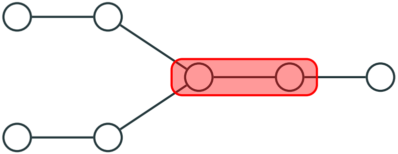
Treewidth by example:



Treewidth

We have shown tractability of PQE on **trees**; let us extend to **bounded treewidth**

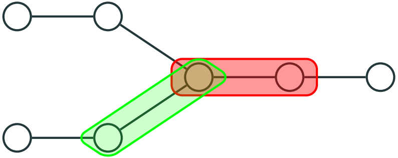
Treewidth by example:



Treewidth

We have shown tractability of PQE on **trees**; let us extend to **bounded treewidth**

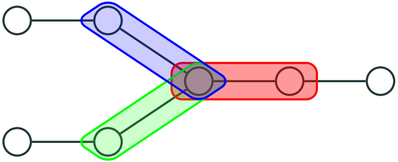
Treewidth by example:



Treewidth

We have shown tractability of PQE on **trees**; let us extend to **bounded treewidth**

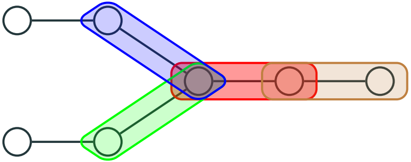
Treewidth by example:



Treewidth

We have shown tractability of PQE on **trees**; let us extend to **bounded treewidth**

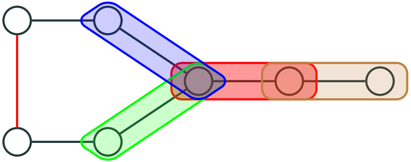
Treewidth by example:



Treewidth

We have shown tractability of PQE on **trees**; let us extend to **bounded treewidth**

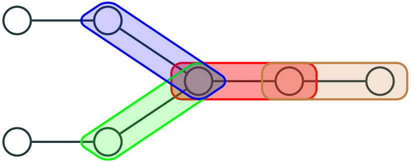
Treewidth by example:

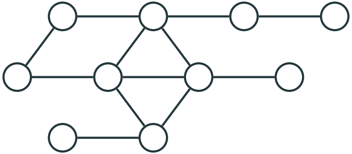


Treewidth

We have shown tractability of PQE on **trees**; let us extend to **bounded treewidth**

Treewidth by example:

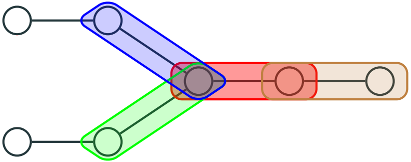


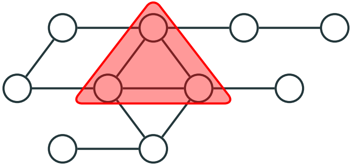


Treewidth

We have shown tractability of PQE on **trees**; let us extend to **bounded treewidth**

Treewidth by example:





Treewidth

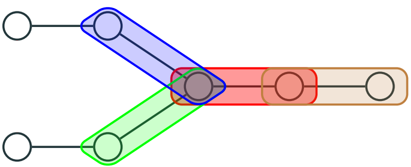
We have shown tractability of PQE on **trees**; let us extend to **bounded treewidth**

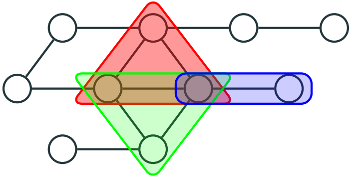
Treewidth by example:

Treewidth

We have shown tractability of PQE on **trees**; let us extend to **bounded treewidth**

Treewidth by example:





Treewidth

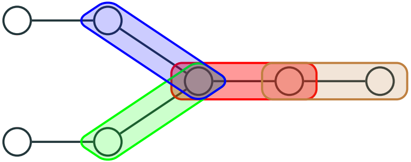
We have shown tractability of PQE on **trees**; let us extend to **bounded treewidth**

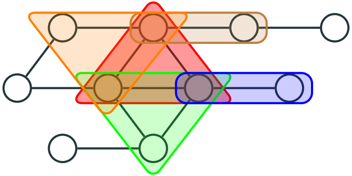
Treewidth by example:

Treewidth

We have shown tractability of PQE on **trees**; let us extend to **bounded treewidth**

Treewidth by example:

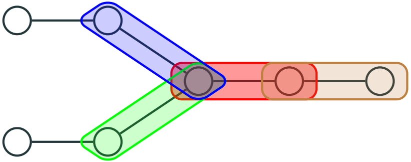


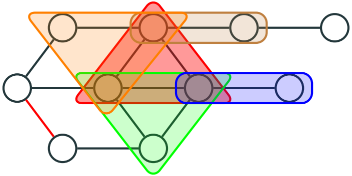


Treewidth

We have shown tractability of PQE on **trees**; let us extend to **bounded treewidth**

Treewidth by example:

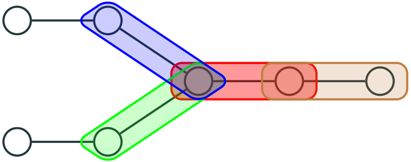


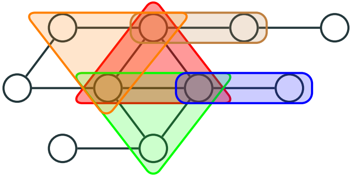


Treewidth

We have shown tractability of PQE on **trees**; let us extend to **bounded treewidth**

Treewidth by example:

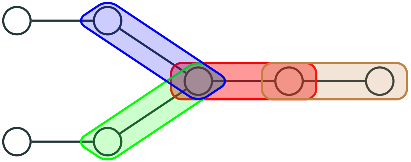


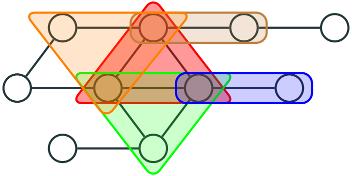


Treewidth

We have shown tractability of PQE on **trees**; let us extend to **bounded treewidth**

Treewidth by example:



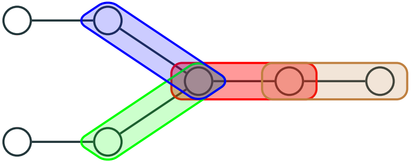


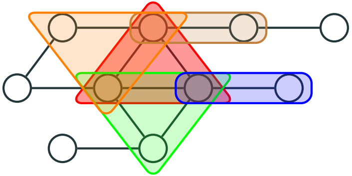
- **Trees** have treewidth 1
- **Cycles** have treewidth 2
- **k-cliques** and **$(k-1)$-grids** have treewidth $k-1$

Treewidth

We have shown tractability of PQE on **trees**; let us extend to **bounded treewidth**

Treewidth by example:



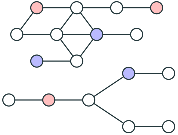


- **Trees** have treewidth **1**
- **Cycles** have treewidth **2**
- **k-cliques** and **$(k-1)$-grids** have treewidth **$k-1$**

→ **Treelike**: the **treewidth** is bounded by a **constant**

Courcelle's theorem and extension to PQE

Treelike **data**

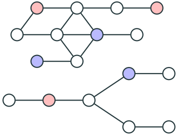


MSO query

$$\exists x\, y$$
$$P_{\text{🔴}}(x) \land P_{\text{🔵}}(y)$$

Courcelle's theorem and extension to PQE

Treelike **data**

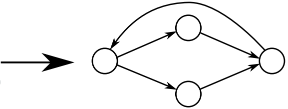


MSO query

$\exists x\, y$

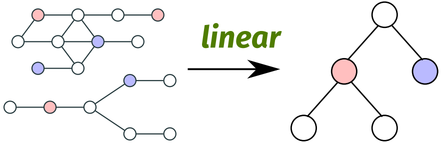
$P_{🔴}(x) \land P_{🔵}(y)$

Tree **automaton**



Courcelle's theorem and extension to PQE

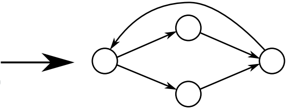
Treelike **data** — Tree **encoding**



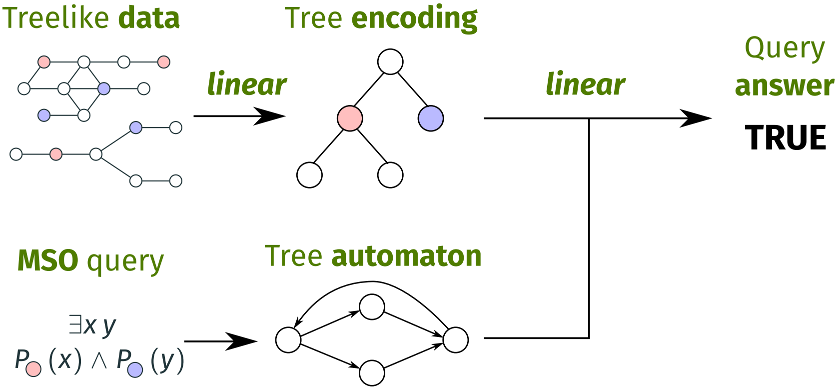
MSO query — Tree **automaton**

$\exists x\, y$

P🔴$(x) \wedge P$🔵(y)



Courcelle's theorem and extension to PQE



Treelike **data**

linear

Tree **encoding**

linear

Query **answer**

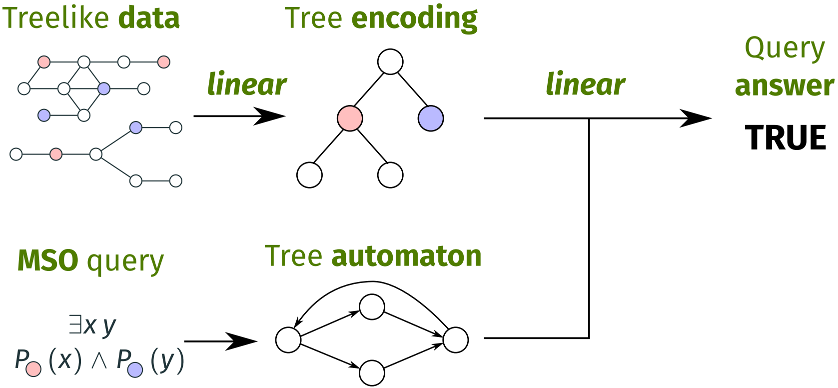
TRUE

MSO query

$$\exists x\, y$$
$$P_{\bullet}(x) \land P_{\bullet}(y)$$

Tree **automaton**

Courcelle's theorem and extension to PQE



Treelike **data**

linear

Tree **encoding**

linear

Query **answer**

TRUE

MSO query

$\exists x\, y$

$P_{\bullet}(x) \wedge P_{\bullet}(y)$

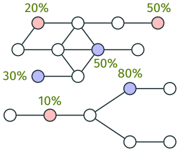
Tree **automaton**

Theorem ([Courcelle, 1990])

For any fixed Boolean MSO query $\boldsymbol{Q}$ *and* $\boldsymbol{k} \in \mathbb{N}$, *given a database* $\boldsymbol{D}$ *of treewidth* $\leq \boldsymbol{k}$, *we can compute in linear time in* $\boldsymbol{D}$ *whether* $\boldsymbol{D}$ *satisfies* $\boldsymbol{Q}$

Courcelle's theorem and extension to PQE

Probabilistic treelike **data**

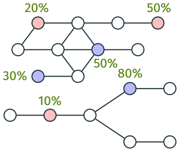


MSO query

$$\exists x\, y$$
$$P_{🔴}(x) \land P_{🔵}(y)$$

Courcelle’s theorem and extension to PQE

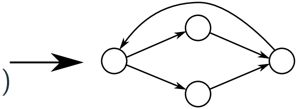
Probabilistic treelike **data**



MSO query

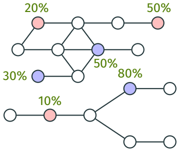
$$\exists x\, y$$
$$P_{\text{🔴}}(x) \wedge P_{\text{🔵}}(y)$$

Tree **automaton**



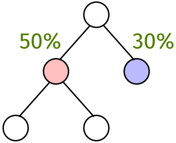
Courcelle’s theorem and extension to PQE

Probabilistic
treelike **data**



linear →

Probabilistic
tree **encoding**

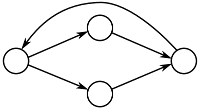


MSO query

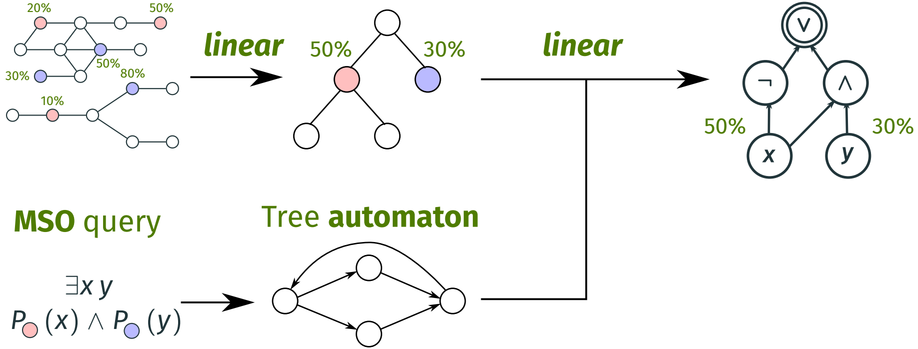
$$\exists x\, y \; P_{\bullet}(x) \wedge P_{\bullet}(y)$$

→

Tree **automaton**



Courcelle's theorem and extension to PQE



Probabilistic treelike **data**

Probabilistic tree **encoding**

d-DNNF circuit with probabilities

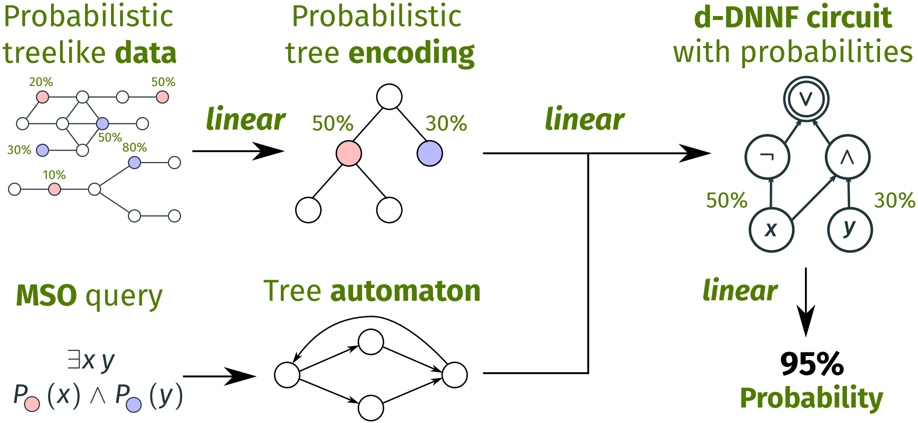
MSO query

$\exists x\, y$

P🔴$(x) \land P$🔵(y)

Tree **automaton**

Courcelle’s theorem and extension to PQE



Probabilistic treelike **data**

linear

Probabilistic tree **encoding**

linear

d-DNNF circuit with probabilities

MSO query

$\exists x\, y$

$P_{🔴}(x) \wedge P_{🔵}(y)$

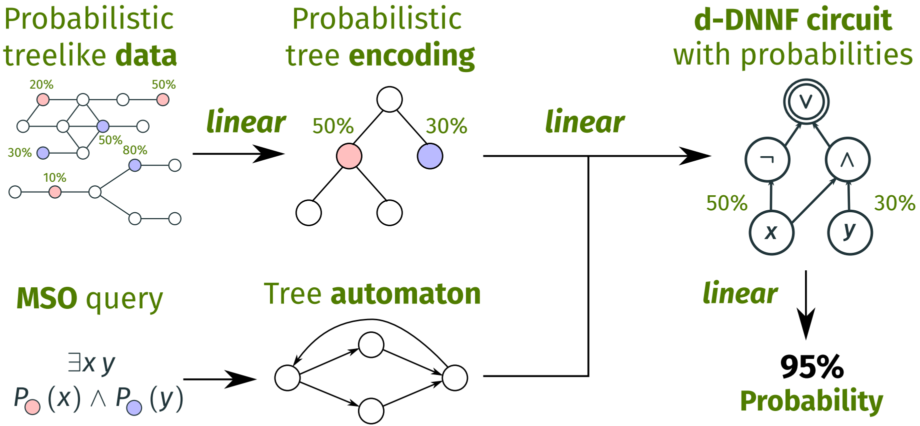
Tree **automaton**

linear

95%

Probability

Courcelle's theorem and extension to PQE



Probabilistic treelike **data**

Probabilistic tree **encoding**

d-DNNF circuit with probabilities

MSO query

$\exists x\, y$
$P_{\bullet}(x) \wedge P_{\bullet}(y)$

Tree **automaton**

95%
Probability

Theorem (A., Bourhis, Senellart, 2015, 2016)

For any fixed Boolean MSO query $\mathbf{Q}$ *and* $\mathbf{k} \in \mathbb{N}$, *given a database* $\mathbf{D}$ *of treewidth* $\leq \mathbf{k}$, *we can solve the PQE problem in linear time (assuming constant-time arithmetics)*

Why is this a dichotomy? Where’s the lower bound?

Why is this a dichotomy? Where's the lower bound?

Theorem (A., Bourhis, Senellart, 2016)

For any arity-two signature, there is a first-order query Q *such that*
for any constructible unbounded-treewidth family $\mathcal{I}$ *of probabilistic graphs,*
the PQE problem for Q *and* $\mathcal{I}$ *is #P-hard under RP reductions*

Why is this a dichotomy? Where's the lower bound?

Theorem (A., Bourhis, Senellart, 2016)

For any arity-two signature, there is a first-order query Q *such that*
for any constructible unbounded-treewidth family $\mathcal{I}$ *of probabilistic graphs,*
the PQE problem for Q *and* $\mathcal{I}$ *is #P-hard under RP reductions*

- **Family:** an infinite set of graphs allowed as input (with arbitrary probabilities) so in particular **closed under subgraphs**

Why is this a dichotomy? Where's the lower bound?

Theorem (A., Bourhis, Senellart, 2016)

For any arity-two signature, there is a ***first-order*** *query* $\mathbf{Q}$ *such that for any constructible* ***unbounded-treewidth*** *family* $\mathcal{I}$ *of probabilistic graphs, the PQE problem for* $\mathbf{Q}$ *and* $\mathcal{I}$ *is* ***#P-hard*** *under RP reductions*

- **Family:** an infinite set of graphs allowed as input (with arbitrary probabilities) so in particular **closed under subgraphs**
- **Unbounded-treewidth**: for all $k \in \mathbb{N}$, there is $I_k \in \mathcal{I}$ of treewidth $\geq k$

Why is this a dichotomy? Where's the lower bound?

Theorem (A., Bourhis, Senellart, 2016)

*For any arity-two signature, there is a **first-order** query* Q *such that*
*for any constructible **unbounded-treewidth** family* $\mathcal{I}$ *of probabilistic graphs,*
the PQE problem for Q *and* $\mathcal{I}$ *is **#P-hard** under RP reductions*

- **Family:** an infinite set of graphs allowed as input (with arbitrary probabilities) so in particular **closed under subgraphs**
- **Unbounded-treewidth**: for all $k \in \mathbb{N}$, there is $I_k \in \mathcal{I}$ of treewidth $\geq k$
- **Constructible:** given k, we can **compute** such an instance I_k in PTIME

Why is this a dichotomy? Where's the lower bound?

Theorem (A., Bourhis, Senellart, 2016)

For any arity-two signature, there is a first-order query Q *such that for any constructible unbounded-treewidth family* $\mathcal{I}$ *of probabilistic graphs, the PQE problem for* Q *and* $\mathcal{I}$ *is #P-hard under RP reductions*

- **Family:** an infinite set of graphs allowed as input (with arbitrary probabilities) so in particular **closed under subgraphs**
- **Unbounded-treewidth**: for all $k \in \mathbb{N}$, there is $I_k \in \mathcal{I}$ of treewidth $\geq k$
- **Constructible:** given k, we can **compute** such an instance I_k in PTIME
- **Under RP reductions:** reduce in PTIME with high probability

Why is this a dichotomy? Where's the lower bound?

Theorem (A., Bourhis, Senellart, 2016)

*For any arity-two signature, there is a **first-order** query **Q** such that for any constructible **unbounded-treewidth** family $\mathcal{I}$ of probabilistic graphs, the PQE problem for **Q** and $\mathcal{I}$ is **#P-hard** under RP reductions*

- **Family:** an infinite set of graphs allowed as input (with arbitrary probabilities) so in particular **closed under subgraphs**
- **Unbounded-treewidth**: for all $k \in \mathbb{N}$, there is $I_k \in \mathcal{I}$ of treewidth $\geq k$
- **Constructible:** given k, we can **compute** such an instance I_k in PTIME
- **Under RP reductions:** reduce in PTIME with high probability

→ This result does **not** generalize to higher-arity!

Why is this a dichotomy? Where's the lower bound?

Theorem (A., Bourhis, Senellart, 2016)

For any arity-two signature, there is a first-order query $\mathbf{Q}$ *such that for any constructible unbounded-treewidth family* $\mathcal{I}$ *of probabilistic graphs, the PQE problem for* $\mathbf{Q}$ *and* $\mathcal{I}$ *is #P-hard under RP reductions*

- **Family:** an infinite set of graphs allowed as input (with arbitrary probabilities) so in particular **closed under subgraphs**
- **Unbounded-treewidth**: for all $k \in \mathbb{N}$, there is $I_k \in \mathcal{I}$ of treewidth $\geq k$
- **Constructible:** given k, we can **compute** such an instance I_k in PTIME
- **Under RP reductions:** reduce in PTIME with high probability

→ This result does **not** generalize to higher-arity!

→ Proof idea: **extract wall graphs as topological minors** ([Chekuri and Chuzhoy, 2014]) and use them for a lower bound

References i

Amarilli, A., Bourhis, P., and Senellart, P. (2015).
Provenance circuits for trees and treelike instances.
In *ICALP*.

Amarilli, A., Bourhis, P., and Senellart, P. (2016).
Tractable lineages on treelike instances: Limits and extensions.
In *PODS*.

Capelli, F. and Mengel, S. (2019).
Tractable QBF by knowledge compilation.
In *STACS*.

References ii

Chekuri, C. and Chuzhoy, J. (2014).
Polynomial bounds for the grid-minor theorem.
In *STOC*.

Courcelle, B. (1990).
The monadic second-order logic of graphs. I. Recognizable sets of finite graphs.
Inf. Comput., 85(1).

Thatcher, J. W. and Wright, J. B. (1968).
Generalized finite automata theory with an application to a decision problem of second-order logic.
Mathematical systems theory, 2(1).

Probabilistic Databases: Other Topics and Conclusion

EDBT-Intended Summer School

Antoine Amarilli

Table of contents

Recursive and homomorphism-closed queries

Going to more general queries

*The case of **UCQs** is settled! But what about **more expressive queries**?*

Going to more general queries

The case of UCQs is settled! But what about more expressive queries?

- Work by [Fink and Olteanu, 2016] about negation
- Some work on ontology-mediated query answering ([Jung and Lutz, 2012])

Going to more general queries

The case of UCQs is settled! But what about more expressive queries?

- Work by [Fink and Olteanu, 2016] about negation
- Some work on ontology-mediated query answering ([Jung and Lutz, 2012])

We study the case of queries closed under homomorphisms

Going to more general queries

*The case of **UCQs** is settled! But what about **more expressive queries**?*

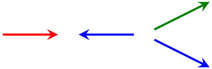
- Work by [Fink and Olteanu, 2016] about **negation**
- Some work on **ontology-mediated query answering** ([Jung and Lutz, 2012])

We study the case of **queries closed under homomorphisms**

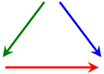
→ We restrict to **arity-two signatures** (work in progress…)

Homomorphism-closed queries

- A **homomorphism** from a graph G to a graph G' maps the vertices of G to those of G' while preserving the edges



has a homomorphism to



Homomorphism-closed queries

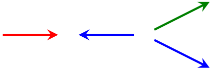
- A homomorphism from a graph G to a graph G' maps the vertices of G to those of G' while preserving the edges


has a homomorphism to

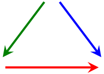
- Homomorphism-closed query Q: for any graph G, if G satisfies Q and G has a homomorphism to G' then G' also satisfies Q

Homomorphism-closed queries

- A **homomorphism** from a graph G to a graph G' maps the vertices of G to those of G' while preserving the edges



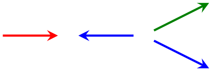
has a homomorphism to



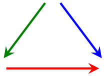
- **Homomorphism-closed query** Q: for any graph G, if G satisfies Q and G has a homomorphism to G' then G' also satisfies Q
- Homomorphism-closed queries include **all CQs**, **all UCQs**, some **recursive queries** like **regular path queries** (RPQs), **Datalog**, etc.

Homomorphism-closed queries

- A **homomorphism** from a graph G to a graph G' maps the vertices of G to those of G' while preserving the edges



has a homomorphism to



- **Homomorphism-closed query** Q: for any graph G, if G satisfies Q and G has a homomorphism to G' then G' also satisfies Q
- Homomorphism-closed queries include **all CQs**, **all UCQs**, some **recursive queries** like **regular path queries** (RPQs), **Datalog**, etc.
- Queries with **negations** or **inequalities** are not homomorphism-closed

Homomorphism-closed queries

- A **homomorphism** from a graph G to a graph G' maps the vertices of G to those of G' while preserving the edges

- **Homomorphism-closed query** Q: for any graph G, if G satisfies Q and G has a homomorphism to G' then G' also satisfies Q
- Homomorphism-closed queries include **all CQs**, **all UCQs**, some **recursive queries** like **regular path queries** (RPQs), **Datalog**, etc.
- Queries with **negations** or **inequalities** are not homomorphism-closed
- Homomorphism-closed queries can equivalently be seen as **infinite unions of CQs** (corresponding to their models)

Our result

We show:

Theorem (A., Ceylan, 2020)

For any query Q closed under homomorphisms on an arity-two signature:

- *Either* Q *is equivalent to a tractable UCQ and* $\mathrm{PQE}(Q)$ *is in PTIME*
- *In all other cases,* $\mathrm{PQE}(Q)$ *is #P-hard*

Our result

We show:

Theorem (A., Ceylan, 2020)

For any query Q closed under homomorphisms on an arity-two signature:

- *Either Q is equivalent to a tractable UCQ and $\mathrm{PQE}(Q)$ is in PTIME*
- *In all other cases, $\mathrm{PQE}(Q)$ is #P-hard*

- The same holds for RPQs, Datalog queries, etc.

Our result

We show:

Theorem (A., Ceylan, 2020)

For any query Q closed under homomorphisms on an arity-two signature:

- *Either* Q *is equivalent to a tractable UCQ and* $\mathrm{PQE}(Q)$ *is in PTIME*
- *In all other cases,* $\mathrm{PQE}(Q)$ *is #P-hard*

- The same holds for RPQs, Datalog queries, etc.
- Example: the RPQ Q: $\longrightarrow (\longrightarrow)^{*} \longrightarrow$

Our result

We show:

Theorem (A., Ceylan, 2020)

*For any **query Q closed under homomorphisms** on an arity-two signature:*

- *Either **Q** is equivalent to a **tractable UCQ** and $\mathrm{PQE}(Q)$ is in **PTIME***
- *In all other cases, $\mathrm{PQE}(Q)$ is **#P-hard***

- The same holds for RPQs, Datalog queries, etc.
- Example: the **RPQ** Q: $\longrightarrow \left(\longrightarrow\right)^{*} \longrightarrow$
  - It is **not equivalent to a UCQ**: infinite disjunction $\longrightarrow \left(\longrightarrow\right)^{i} \longrightarrow$ for all $i \in \mathbb{N}$

Our result

We show:

Theorem (A., Ceylan, 2020)

For any query Q closed under homomorphisms on an arity-two signature:

- *Either Q is equivalent to a tractable UCQ and $\mathrm{PQE}(Q)$ is in PTIME*
- *In all other cases, $\mathrm{PQE}(Q)$ is #P-hard*

- The same holds for RPQs, Datalog queries, etc.
- Example: the RPQ Q: $\longrightarrow \left(\longrightarrow\right)^{*} \longrightarrow$
  - It is not equivalent to a UCQ: infinite disjunction $\longrightarrow \left(\longrightarrow\right)^{i} \longrightarrow$ for all $i \in \mathbb{N}$
  - Hence, $\mathrm{PQE}(Q)$ is #P-hard

Uniform probabilities

Uniform probabilities: Problem statement

What if we restricted probabilities on input instances to always be $1/2$*?*

Uniform probabilities: Problem statement

What if we restricted probabilities on input instances to always be $1/2$*?*

- The PQE problem becomes the **uniform reliability** (UR) problem:
  - $\rightarrow$ $\mathrm{UR}(Q)$: given a graph, how many of its subgraphs satisfy Q

Uniform probabilities: Problem statement

What if we restricted probabilities on input instances to always be $1/2$*?*

- The PQE problem becomes the **uniform reliability** (UR) problem:
  - → $\mathrm{UR}(Q)$: given a graph, how many of its subgraphs satisfy Q
- The UR problem **reduces** to PQE, but no obvious reduction in the other direction

Uniform probabilities: Problem statement

What if we restricted probabilities on input instances to always be $1/2$?

- The PQE problem becomes the **uniform reliability** (UR) problem:
  - → $\mathrm{UR}(Q)$: given a graph, how many of its subgraphs satisfy Q
- The UR problem **reduces** to PQE, but no obvious reduction in the other direction

We limit to **self-join-free CQs** and extend the "small" Dalvi and Suciu dichotomy to UR:

Theorem (A., Kimelfeld, 2022)

Let Q be a self-join-free CQ:

- *If Q is hierarchical, then $\mathrm{PQE}(Q)$ is in PTIME*
- *Otherwise, even $\mathrm{UR}(Q)$ is #P-hard*

Approximate evaluation

Approximation

- When it’s too hard to compute the exact probability, we can approximate it

Approximation

- When it's too hard to compute the exact probability, we can approximate it
- One possibility is to compute a lower bound and upper bound:
  - $\max(\Pr(\phi), \Pr(\psi)) \leq \Pr(\phi \vee \psi) \leq \min(\Pr(\phi) + \Pr(\psi), 1)$
  - $\max(0, \Pr(\phi) + \Pr(\psi) - 1) \leq \Pr(\phi \wedge \psi) \leq \min(\Pr(\phi), \Pr(\psi))$ (by duality)
  - $\Pr(\neg\phi) = 1 - \Pr(\phi)$ (reminder)

Approximation by sampling

Another possibility is to approximate via Monte-Carlo sampling:

- Pick a random possible world according to the fact probabilities:
  - → Keep F with probability $\Pr(F)$ and discard it otherwise
  - → Repeat for the other variables

Approximation by sampling

Another possibility is to approximate via **Monte-Carlo sampling**:

- Pick a random **possible world** according to the fact probabilities:
  - → Keep F with probability $\Pr(F)$ and discard it otherwise
  - → Repeat for the other variables
- **Evaluate** the lineage formula ϕ under this valuation

Approximation by sampling

Another possibility is to approximate via **Monte-Carlo sampling**:

- Pick a random **possible world** according to the fact probabilities:
  - → Keep F with probability $\Pr(F)$ and discard it otherwise
  - → Repeat for the other variables
- **Evaluate** the lineage formula ϕ under this valuation
- Approximate the probability of the formula ϕ as the **proportion of times** when it was true

Approximation by sampling

Another possibility is to approximate via **Monte-Carlo sampling**:

- Pick a random **possible world** according to the fact probabilities:
  - → Keep F with probability $\Pr(F)$ and discard it otherwise
  - → Repeat for the other variables
- **Evaluate** the lineage formula ϕ under this valuation
- Approximate the probability of the formula ϕ as the **proportion of times** when it was true
- **Theoretical guarantees:** on how many samples suffice so that, with high probability, the estimated probability is almost correct

Other method for a **multiplicative approximation**: Karp-Luby algorithm

Using external tools

- Specialized software to compute the probability of a formula: **weighted model counters**
- Examples (ongoing research):
    - **c2d:** `http://reasoning.cs.ucla.edu/c2d/download.php`
    - **d4:** `https://www.cril.univ-artois.fr/KC/d4.html`
    - **dsharp:** `https://bitbucket.org/haz/dsharp`

Repairs

Repairs

- Another kind of uncertainty: we know that the database must satisfy some **constraints** (e.g., functionality)
- The data that we have does **not** satisfy it
- Reason about the ways to **repair** the data, e.g., removing a minimal subset of tuples
- Can we **evaluate queries** on this representation? E.g., is a query true on **every maximal repair**? See, e.g., [Koutris and Wijsen, 2015].

→ Tutorial by Jef Wijsen

Incompleteness: Open-World Query Answering

Open-world query answering

- Most data sources are **incomplete**, e.g., Wikidata
- **Idea:** see an incomplete data source as representing **all possible completions**
- A query result is **certain** if it is true on **every possible completion**
- We also assume **constraints** to restrict the possible completions (e.g., IDs and FDs, see Andreas's talk)

Open-world query answering problem

Definition of the **open-world query answering** problem (OWQA):

- Given:
  - An incomplete **database** D
  - Logical **constraints** Σ on the true state of the world
  - A **query** Q
- Determine if Q is true in **every completion** of D that satisfies Σ

Open-world query answering problem

Definition of the open-world query answering problem (OWQA):

- Given:
  - An incomplete database D
  - Logical constraints Σ on the true state of the world
  - A query Q
- Determine if Q is true in every completion of D that satisfies Σ
- Equivalently: satisfiability of $D \land \Sigma \land \neg Q$

Open-world query answering problem

Definition of the open-world query answering problem (OWQA):

- Given:
  - An incomplete database D
  - Logical constraints Σ on the true state of the world
  - A query Q
- Determine if Q is true in every completion of D that satisfies Σ
- Equivalently: satisfiability of $D \land \Sigma \land \neg Q$

Note: We assume that the incomplete database D satisfies the constraints.
(Otherwise we need to repair it.)

Results on OWQA

- The OWQA problem can be **undecidable** if we allow **arbitrary first-order logic** for Σ
- It is also undecidable for **common database constraint languages**, e.g., tuple-generating dependencies
- It is **decidable** for **better-behaved** logical fragments, e.g., the **guarded fragment**

Results on OWQA

- The OWQA problem can be **undecidable** if we allow **arbitrary first-order logic** for Σ
- It is also undecidable for **common database constraint languages**, e.g., tuple-generating dependencies
- It is **decidable** for **better-behaved** logical fragments, e.g., the **guarded fragment**
- Two main techniques:
  - **Forward chaining**, aka the “chase”: add data to satisfy the constraints:

Results on OWQA

- The OWQA problem can be **undecidable** if we allow **arbitrary first-order logic** for Σ
- It is also undecidable for **common database constraint languages**, e.g., tuple-generating dependencies
- It is **decidable** for **better-behaved** logical fragments, e.g., the **guarded fragment**
- Two main techniques:
  - **Forward chaining**, aka the “chase”: add data to satisfy the constraints:
    - If the process **terminates**, use the result to satisfy the query

Results on OWQA

- The OWQA problem can be **undecidable** if we allow **arbitrary first-order logic** for Σ
- It is also undecidable for **common database constraint languages**, e.g., tuple-generating dependencies
- It is **decidable** for **better-behaved** logical fragments, e.g., the **guarded fragment**
- Two main techniques:
  - **Forward chaining**, aka the “chase”: add data to satisfy the constraints:
    - If the process **terminates**, use the result to satisfy the query
    - If it is infinite but has bounded **treewidth**, reason over it, e.g., with automata

Results on OWQA

- The OWQA problem can be **undecidable** if we allow **arbitrary first-order logic** for Σ
- It is also undecidable for **common database constraint languages**, e.g., tuple-generating dependencies
- It is **decidable** for **better-behaved** logical fragments, e.g., the **guarded fragment**
- Two main techniques:
  - **Forward chaining**, aka the "chase": add data to satisfy the constraints:
    - If the process **terminates**, use the result to satisfy the query
    - If it is infinite but has bounded **treewidth**, reason over it, e.g., with automata
  - **Backward chaining**, aka "query rewriting": change the query to reflect the constraints

Incompleteness: NULLs

Codd tables, a.k.a. SQL NULLs

Patient	Examin. 1	Examin. 2	Diagnosis
A	23	12	α
B	10	23	$\perp_1$
C	2	4	γ
D	15	15	$\perp_2$
E	$\perp_3$	17	β

- Most **simple** form of incomplete database
- **Widely used** in practice, in DBMS since the mid-1970s!
- All NULLs ($\perp$) are considered **distinct**
- Possible world semantics: all possible completions of the table (infinitely many)
- In SQL, **three-valued logic**, weird semantics:

```
SELECT * FROM Tel WHERE tel_nr = '333' OR tel_nr <> '333'
```

Problem: Codd tables and query evaluation

Appointment

Doctor	Patient
D1	A
D2	A

Illness

Patient	Diagnosis
A	$\perp$

Let's join the two tables...

Problem: Codd tables and query evaluation

Appointment

Doctor	Patient
D1	A
D2	A

Illness

Patient	Diagnosis
A	$\bot$

Let's join the two tables...

Appointment $\bowtie$ Illness

Doctor	Patient	Diagnosis

Problem: Codd tables and query evaluation

Appointment

Doctor	Patient
D1	A
D2	A

Illness

Patient	Diagnosis
A	$\perp$

Let's **join** the two tables...

Appointment $\bowtie$ Illness

Doctor	Patient	Diagnosis
D1	A	$\perp_1$
D2	A	$\perp_2$

Problem: Codd tables and query evaluation

Appointment

Doctor	Patient
D1	A
D2	A

Illness

Patient	Diagnosis
A	$\perp$

Let’s **join** the two tables...

Appointment $\bowtie$ Illness

Doctor	Patient	Diagnosis
D1	A	$\perp_1$
D2	A	$\perp_2$

- We know that $\perp_1 = \perp_2$, but we cannot **represent it**
- Simple solution: **named nulls** aka v-tables
- More expressive solution: **c-tables**

Summary and directions

Summary of what we have seen

- Probabilistic database model: TIDs, facts have independent probabilities
  - → Also more expressive models: BIDs, pc-tables

Summary of what we have seen

- Probabilistic database model: **TIDs**, facts have independent probabilities
  - → Also more expressive models: BIDs, pc-tables
- **Probabilistic query evaluation** (PQE) for queries on probabilistic databases
  - → Research question: for which queries is PQE tractable?

Summary of what we have seen

- Probabilistic database model: **TIDs**, facts have independent probabilities
  - → Also more expressive models: BIDs, pc-tables
- **Probabilistic query evaluation** (PQE) for queries on probabilistic databases
  - → Research question: for which queries is PQE tractable?
- Dichotomy on **self-join free CQs**: PQE is tractable precisely for hierarchical queries
  - → Extends to a more complex dichotomy on UCQs

Summary of what we have seen

- Probabilistic database model: **TIDs**, facts have independent probabilities
  - → Also more expressive models: BIDs, pc-tables
- **Probabilistic query evaluation** (PQE) for queries on probabilistic databases
  - → Research question: for which queries is PQE tractable?
- Dichotomy on **self-join free CQs**: PQE is tractable precisely for hierarchical queries
  - → Extends to a more complex dichotomy on UCQs
- We can solve PQE for hierarchical self-join-free CQs with **d-DNNF circuits**
  - → This is open for UCQs: **intensional–extensional conjecture**

Summary of what we have seen

- Probabilistic database model: **TIDs**, facts have independent probabilities
  - → Also more expressive models: BIDs, pc-tables
- **Probabilistic query evaluation** (PQE) for queries on probabilistic databases
  - → Research question: for which queries is PQE tractable?
- Dichotomy on **self-join free CQs**: PQE is tractable precisely for hierarchical queries
  - → Extends to a more complex dichotomy on UCQs
- We can solve PQE for hierarchical self-join-free CQs with **d-DNNF circuits**
  - → This is open for UCQs: **intensional–extensional conjecture**
- We can make **all queries in MSO** tractable by bounding the instance **treewidth**
  - → And MSO is intractable if you do not bound treewidth (under some conditions)

Summary of what we have seen

- Probabilistic database model: **TIDs**, facts have independent probabilities
  - → Also more expressive models: BIDs, pc-tables
- **Probabilistic query evaluation** (PQE) for queries on probabilistic databases
  - → Research question: for which queries is PQE tractable?
- Dichotomy on **self-join free CQs**: PQE is tractable precisely for hierarchical queries
  - → Extends to a more complex dichotomy on UCQs
- We can solve PQE for hierarchical self-join-free CQs with **d-DNNF circuits**
  - → This is open for UCQs: **intensional–extensional conjecture**
- We can make **all queries in MSO** tractable by bounding the instance **treewidth**
  - → And MSO is intractable if you do not bound treewidth (under some conditions)
- Extensions: homomorphism-closed queries, uniform reliability...

Other topics of research

- Queries with **negation** [Fink and Olteanu, 2016]
- Queries with **inequalities** [Olteanu and Huang, 2009]
- **Symmetric model counting** [Beame et al., 2015]
- A summary: Dan Suciu, *Probabilistic Databases for All* [Suciu, 2020]

Other topics of research

- Queries with **negation** [Fink and Olteanu, 2016]
- Queries with **inequalities** [Olteanu and Huang, 2009]
- **Symmetric model counting** [Beame et al., 2015]
- A summary: Dan Suciu, *Probabilistic Databases for All* [Suciu, 2020]

And recently:

- **Infinite domains** [Carmeli et al., 2021]
- **PQE under updates** [Berkholz and Merz, 2021]
- **Open-world probabilistic databases** [Ceylan et al., 2021]
- **Active probabilistic databases** [Drien et al., 2022]

Other topics of research

- Queries with **negation** [Fink and Olteanu, 2016]
- Queries with **inequalities** [Olteanu and Huang, 2009]
- **Symmetric model counting** [Beame et al., 2015]
- A summary: Dan Suciu, *Probabilistic Databases for All* [Suciu, 2020]

And recently:

- **Infinite domains** [Carmeli et al., 2021]
- **PQE under updates** [Berkholz and Merz, 2021]
- **Open-world probabilistic databases** [Ceylan et al., 2021]
- **Active probabilistic databases** [Drien et al., 2022]
- (Others? talk to me :))

Future research directions

- Reusability of techniques : repairs (see talk by Jef), Shapley values (see talk by Benny), graphical models, probabilistic programming, probabilistic constraints…

Future research directions

- **Reusability** of techniques : repairs (see talk by Jef), Shapley values (see talk by Benny), graphical models, probabilistic programming, probabilistic constraints...
- Connection to **theoretical research**, e.g., CSP
  - → Conjecture: for any homomorphism-closed query Q, given an instance, the **uniform reliability problem** for Q is either #P-hard or PTIME
  - → Working on **unbounded queries**, UCQ case also **open** [Kenig and Suciu, 2021]

Future research directions

- **Reusability** of techniques : repairs (see talk by Jef), Shapley values (see talk by Benny), graphical models, probabilistic programming, probabilistic constraints...
- Connection to **theoretical research**, e.g., CSP
  - → Conjecture: for any homomorphism-closed query Q, given an instance, the **uniform reliability problem** for Q is either #P-hard or PTIME
  - → Working on **unbounded queries**, UCQ case also **open** [Kenig and Suciu, 2021]
- Practical implementation: **ProvSQL**, but what about aggregates? numerical imprecision? approximations?
  - → Can we compute **multiplicative approximations** for **recursive queries**?

Future research directions

- **Reusability** of techniques : repairs (see talk by Jef), Shapley values (see talk by Benny), graphical models, probabilistic programming, probabilistic constraints...
- Connection to **theoretical research**, e.g., CSP
  - → Conjecture: for any homomorphism-closed query Q, given an instance, the **uniform reliability problem** for Q is either #P-hard or PTIME
  - → Working on **unbounded queries**, UCQ case also **open** [Kenig and Suciu, 2021]
- Practical implementation: **ProvSQL**, but what about aggregates? numerical imprecision? approximations?
  - → Can we compute **multiplicative approximations** for **recursive queries**?
- Connections to knowledge compilation and **intensional–extensional** conjecture
  - → Can we compute the **provenance** of tractable UCQs in a tractable formalism, e.g., **d-Ds**?

Future research directions

- **Reusability** of techniques : repairs (see talk by Jef), Shapley values (see talk by Benny), graphical models, probabilistic programming, probabilistic constraints...

- Connection to **theoretical research**, e.g., CSP
  - → Conjecture: for any homomorphism-closed query Q, given an instance, the **uniform reliability problem** for Q is either #P-hard or PTIME
  - → Working on **unbounded queries**, UCQ case also **open** [Kenig and Suciu, 2021]

- Practical implementation: **ProvSQL**, but what about aggregates? numerical imprecision? approximations?
  - → Can we compute **multiplicative approximations** for **recursive queries**?

- Connections to knowledge compilation and **intensional–extensional** conjecture
  - → Can we compute the **provenance** of tractable UCQs in a tractable formalism, e.g., **d-Ds**?

- Combining the **query-based** and **structure-based** approaches

Future research directions

- **Reusability** of techniques : repairs (see talk by Jef), Shapley values (see talk by Benny), graphical models, probabilistic programming, probabilistic constraints...
- Connection to **theoretical research**, e.g., CSP
  - → Conjecture: for any homomorphism-closed query Q, given an instance, the **uniform reliability problem** for Q is either #P-hard or PTIME
  - → Working on **unbounded queries**, UCQ case also **open** [Kenig and Suciu, 2021]
- Practical implementation: **ProvSQL**, but what about aggregates? numerical imprecision? approximations?
  - → Can we compute **multiplicative approximations** for **recursive queries**?
- Connections to knowledge compilation and **intensional–extensional** conjecture
  - → Can we compute the **provenance** of tractable UCQs in a tractable formalism, e.g., **d-Ds**?
- Combining the **query-based** and **structure-based** approaches

Thanks for your attention!

References i

Abiteboul, S., Kimelfeld, B., Sagiv, Y., and Senellart, P. (2009).
On the expressiveness of probabilistic XML models.
VLDB Journal, 18(5).

Amarilli, A., Bourhis, P., and Senellart, P. (2015).
Provenance circuits for trees and treelike instances.
In *ICALP*.

Amarilli, A., Bourhis, P., and Senellart, P. (2016).
Tractable lineages on treelike instances: Limits and extensions.
In *PODS*.

References ii

Amarilli, A. and Ceylan, I. I. (2020).
A dichotomy for homomorphism-closed queries on probabilistic graphs.
In *ICDT*.

Amarilli, A. and Kimelfeld, B. (2022).
Uniform Reliability of Self-Join-Free Conjunctive Queries.
Under review.

Beame, P., Van den Broeck, G., Gribkoff, E., and Suciu, D. (2015).
Symmetric weighted first-order model counting.
In *PODS*.

Benedikt, M., Kharlamov, E., Olteanu, D., and Senellart, P. (2010).
Probabilistic XML via Markov chains.
PVLDB, 3(1).

References iii

Berkholz, C. and Merz, M. (2021).
Probabilistic databases under updates: Boolean query evaluation and ranked enumeration.
In *PODS*.

Carmeli, N., Grohe, M., Lindner, P., and Standke, C. (2021).
Tuple-independent representations of infinite probabilistic databases.
In *PODS*.

Ceylan, I. I., Darwiche, A., and Van den Broeck, G. (2021).
Open-world probabilistic databases: Semantics, algorithms, complexity.
Artificial Intelligence, 295.

References iv

Cohen, S., Kimelfeld, B., and Sagiv, Y. (2009).
Running tree automata on probabilistic xml.
In *PODS*.

Dalvi, N., Ré, C., and Suciu, D. (2009).
Probabilistic databases: Diamonds in the dirt.
Communications of the ACM, 52(7).

Dalvi, N. N. and Suciu, D. (2004).
Efficient query evaluation on probabilistic databases.
In *VLDB*.

Drien, O., Freiman, M., and Amsterdamer, Y. (2022).
ActivePDB: Active probabilistic databases.
Working draft.

References v

Fink, R. and Olteanu, D. (2016).
Dichotomies for queries with negation in probabilistic databases.
ACM Transactions on Database Systems, 41(1).

Imielinski, T. and Lipski, W. (1984).
Incomplete information in relational databases.
Journal of the ACM, 31(4).

Jung, J. C. and Lutz, C. (2012).
Ontology-based access to probabilistic data with OWL QL.
In *ISWC*.

References vi

Kenig, B. and Suciu, D. (2021).
A dichotomy for the generalized model counting problem for unions of conjunctive queries.
In *PODS*.

Koutris, P. and Wijsen, J. (2015).
The data complexity of consistent query answering for self-join-free conjunctive queries under primary key constraints.
In *SIGMOD*.

Olteanu, D. and Huang, J. (2009).
Secondary-storage confidence computation for conjunctive queries with inequalities.
In *SIGMOD*.

References vii

Suciu, D. (2020).
Probabilistic databases for all.
In *PODS*.

Widom, J. (2005).
Trio: A system for integrated management of data, accuracy, and lineage.
In *Proc. CIDR*.

Basic idea: finding a tight pattern

The challenging part is to show:

Theorem

For any query Q closed under homomorphisms and unbounded, $\mathrm{PQE}(Q)$ is #P-hard

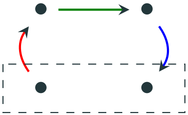
Basic idea: finding a tight pattern

The challenging part is to show:

Theorem

For any query Q closed under homomorphisms and unbounded, $\mathrm{PQE}(Q)$ *is #P-hard*

Idea: find a tight pattern, i.e., a graph with three distinguished edges → → → such that:



satisfies Q

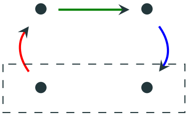
Basic idea: finding a tight pattern

The challenging part is to show:

Theorem

For any query Q closed under homomorphisms and unbounded, $\mathrm{PQE}(Q)$ *is #P-hard*

Idea: find a **tight pattern**, i.e., a graph with three distinguished edges → → → such that:



but

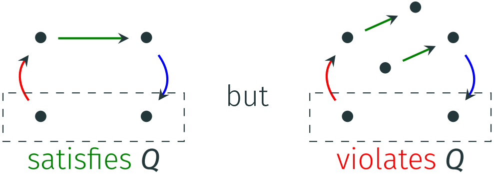
Basic idea: finding a tight pattern

The challenging part is to show:

Theorem

For any query Q closed under homomorphisms and unbounded, $\mathrm{PQE}(Q)$ *is #P-hard*

Idea: find a tight pattern, i.e., a graph with three distinguished edges → → → such that:



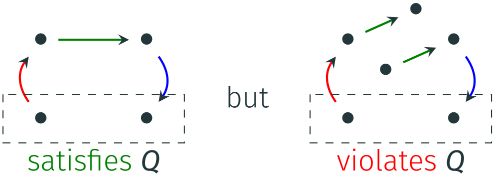
Basic idea: finding a tight pattern

The challenging part is to show:

Theorem

*For any query Q closed under homomorphisms and **unbounded**, $\mathrm{PQE}(Q)$ is **#P-hard***

Idea: find a **tight pattern**, i.e., a graph with three distinguished edges → → → such that:

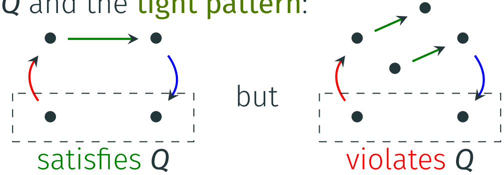


Theorem

Any unbounded query closed under homomorphisms has a tight pattern

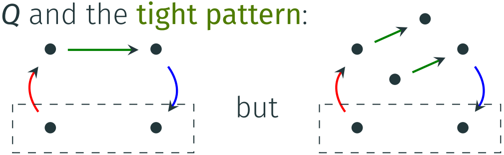
Using tight patterns to show hardness of PQE

- Fix the query Q and the **tight pattern**:



Using tight patterns to show hardness of PQE

- Fix the query Q and the **tight pattern**:

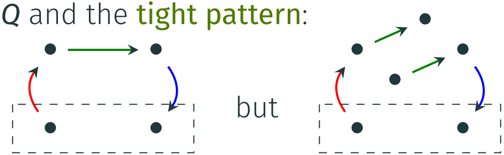


satisfies Q but violates Q

- We reduce from PQE for the **intractable** CQ: $Q_0 : x \longrightarrow y \longrightarrow z \longrightarrow w$

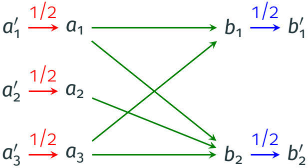
Using tight patterns to show hardness of PQE

- Fix the query Q and the **tight pattern**:

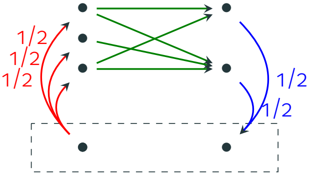


satisfies Q but violates Q

- We reduce from PQE for the **intractable** CQ: $Q_0 : x \xrightarrow{} y \xrightarrow{} z \xrightarrow{} w$

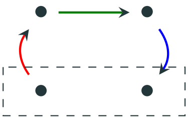


is coded as



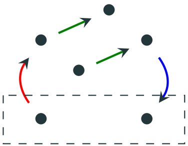
Using tight patterns to show hardness of PQE

- Fix the query Q and the **tight pattern**:



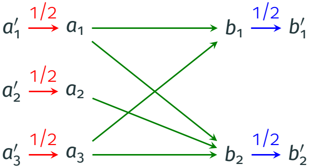
satisfies Q

but

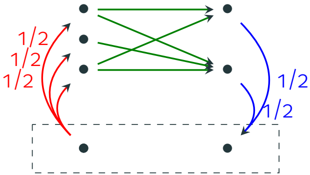


violates Q

- We reduce from PQE for the **intractable** CQ: $Q_0 : x \rightarrow y \rightarrow z \rightarrow w$



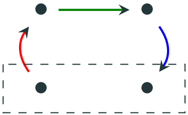
is coded as



Idea: possible worlds at the **left** have a path that matches Q_0
iff the corresponding possible world of the TID at the **right** satisfies the query Q...

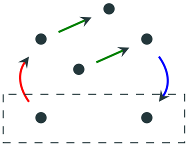
Using tight patterns to show hardness of PQE

- Fix the query Q and the **tight pattern**:



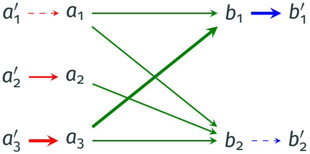
satisfies Q

but

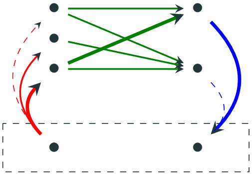


violates Q

- We reduce from PQE for the **intractable** CQ: $Q_0 : x \xrightarrow{} y \xrightarrow{} z \xrightarrow{} w$



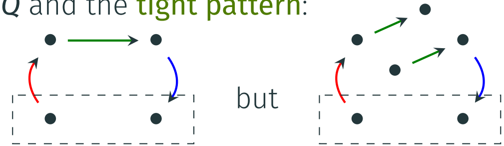
is coded as



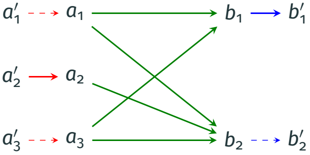
Idea: possible worlds at the **left** have a path that matches Q_0
iff the corresponding possible world of the TID at the **right** satisfies the query Q...

Using tight patterns to show hardness of PQE

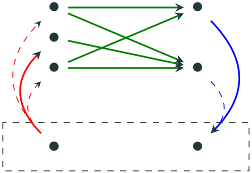
- Fix the query Q and the **tight pattern**:



- We reduce from PQE for the **intractable** CQ: $Q_0 : x \rightarrow y \rightarrow z \rightarrow w$



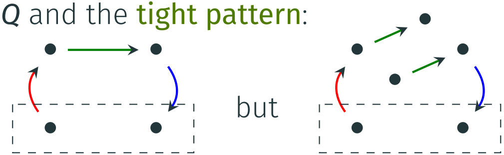
is coded as



Idea: possible worlds at the **left** have a path that matches Q_0
iff the corresponding possible world of the TID at the **right** satisfies the query Q...

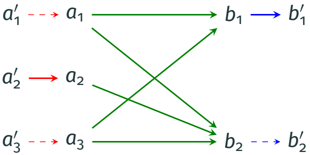
Using tight patterns to show hardness of PQE

- Fix the query Q and the **tight pattern**:

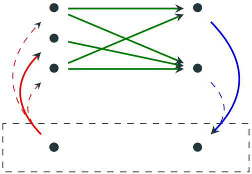


satisfies Q but violates Q

- We reduce from PQE for the **intractable** CQ: $Q_0 : x \longrightarrow y \longrightarrow z \longrightarrow w$



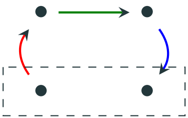
is coded as



Idea: possible worlds at the **left** have a path that matches Q_0
iff the corresponding possible world of the TID at the **right** satisfies the query Q...
... except we need **more** from the tight pattern!

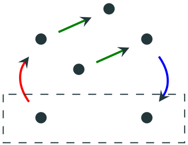
Using tight patterns to show hardness of PQE

- Fix the query *Q* and the **tight pattern**:



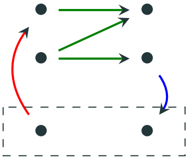
satisfies *Q*

but



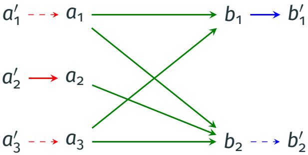
violates *Q*

AND

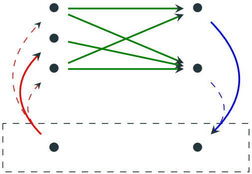


violates *Q*

- We reduce from PQE for the **intractable** CQ: $Q_0 : x \longrightarrow y \longrightarrow z \longrightarrow w$



is coded as



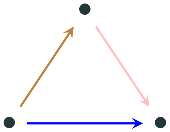
Idea: possible worlds at the **left** have a path that matches Q_0

iff the corresponding possible world of the TID at the **right** satisfies the query *Q*…

… except we need **more** from the tight pattern!

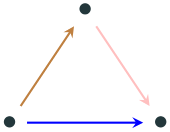
Why can we always find tight patterns?

- Unbounded queries have arbitrarily large minimal models
- Take a large minimal model *D* and disconnect its edges:

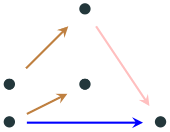


Why can we always find tight patterns?

- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model *D* and **disconnect its edges**:

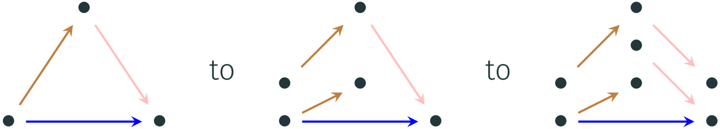


to



Why can we always find tight patterns?

- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model *D* and **disconnect its edges**:

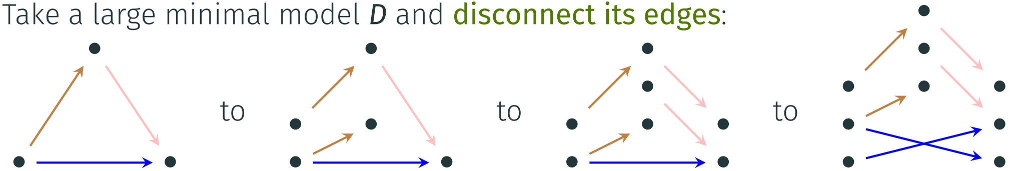


to

to

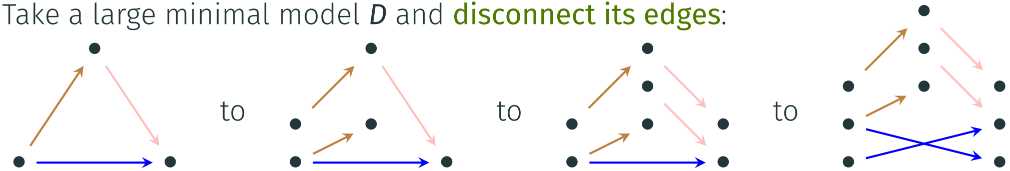
Why can we always find tight patterns?

- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model D and **disconnect its edges**:



Why can we always find tight patterns?

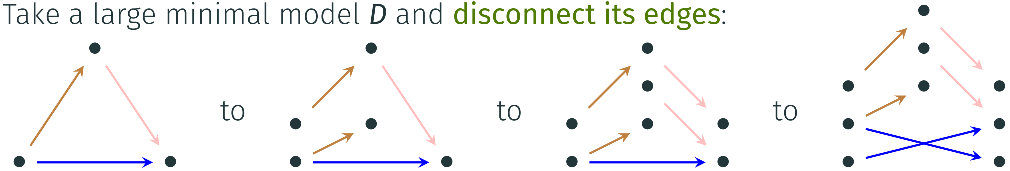
- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model D and **disconnect its edges**:



- If Q becomes false at one step, then we have found a **tight pattern**

Why can we always find tight patterns?

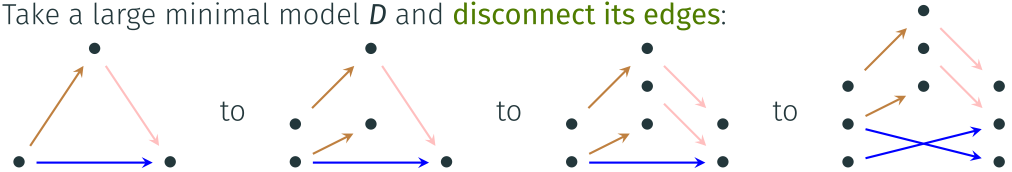
- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model *D* and **disconnect its edges**:



- If *Q* becomes false at one step, then we have found a **tight pattern**
- Otherwise, we have found a **contradiction**:
    - The disconnection process **terminates**

Why can we always find tight patterns?

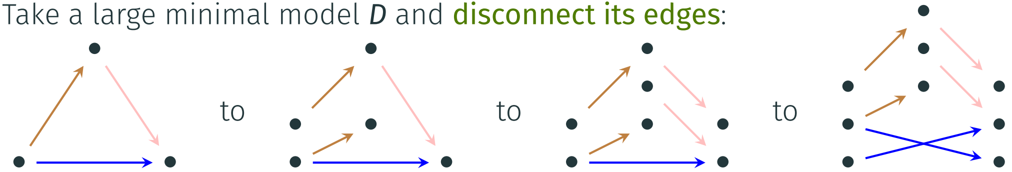
- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model D and **disconnect its edges**:



- If Q becomes false at one step, then we have found a **tight pattern**
- Otherwise, we have found a **contradiction**:
  - The disconnection process **terminates**
  - At the end of the process, we obtain a **star** D'

Why can we always find tight patterns?

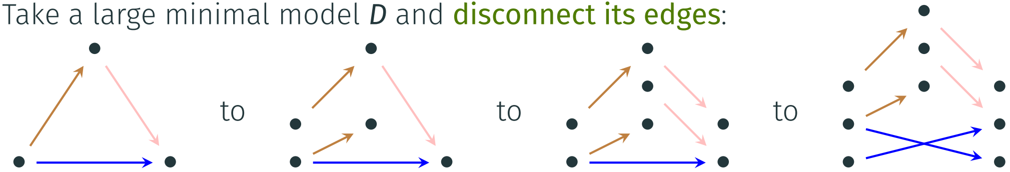
- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model D and **disconnect its edges**:



- If Q becomes false at one step, then we have found a **tight pattern**
- Otherwise, we have found a **contradiction**:
  - The disconnection process **terminates**
  - At the end of the process, we obtain a **star** D'
  - It is **homomorphically equivalent** to a constant-sized D'' satisfying Q

Why can we always find tight patterns?

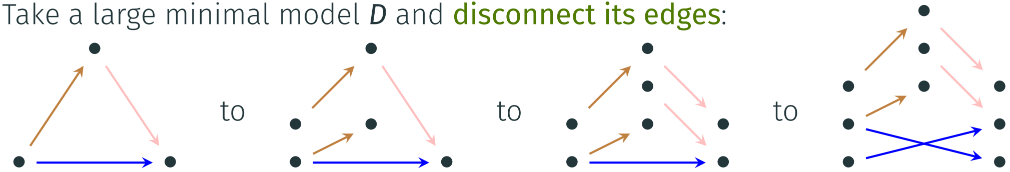
- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model D and **disconnect its edges**:



- If Q becomes false at one step, then we have found a **tight pattern**
- Otherwise, we have found a **contradiction**:
  - The disconnection process **terminates**
  - At the end of the process, we obtain a **star** D'
  - It is **homomorphically equivalent** to a constant-sized D'' satisfying Q
  - D'' has a **homomorphism** back to D

Why can we always find tight patterns?

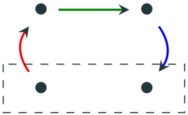
- Unbounded queries have **arbitrarily large** minimal models
- Take a large minimal model *D* and **disconnect its edges**:



- If *Q* becomes false at one step, then we have found a **tight pattern**
- Otherwise, we have found a **contradiction**:
  - The disconnection process **terminates**
  - At the end of the process, we obtain a **star** D'
  - It is **homomorphically equivalent** to a constant-sized D'' satisfying *Q*
  - D'' has a **homomorphism** back to *D*
  - This contradicts the **minimality** of the large *D*

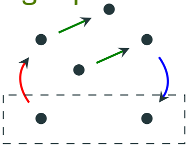
Rescuing the proof

We know that we have a **tight pattern**:



satisfies Q

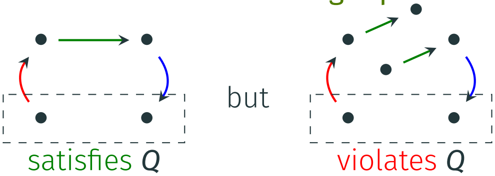
but



violates Q

Rescuing the proof

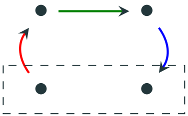
We know that we have a **tight pattern**:



Consider its **iterates**

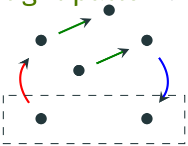
Rescuing the proof

We know that we have a **tight pattern**:



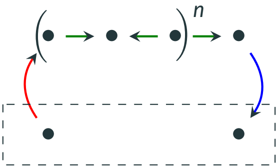
satisfies Q

but



violates Q

Consider its **iterates** for each $n \in \mathbb{N}$:

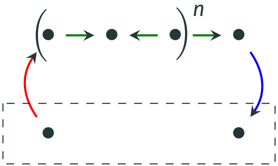


Rescuing the proof

We know that we have a **tight pattern**:

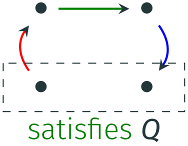
satisfies Q but violates Q

Consider its **iterates** for each $n \in \mathbb{N}$:

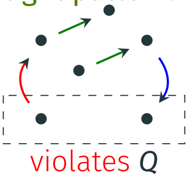


Rescuing the proof

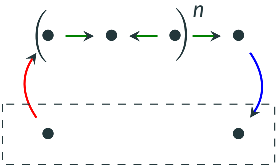
We know that we have a **tight pattern**:



but

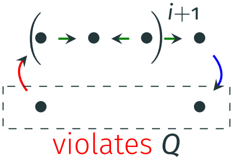


Consider its **iterates** for each $n \in \mathbb{N}$:



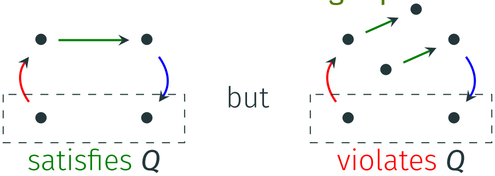
Case 1: some iterate **violates** the query:

but



Rescuing the proof

We know that we have a **tight pattern**:

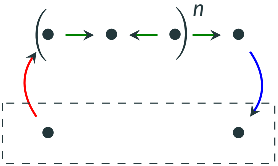


satisfies Q

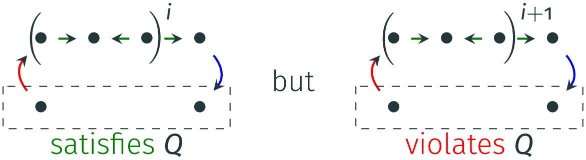
but

violates Q

Consider its **iterates** for each $n \in \mathbb{N}$:



Case 1: some iterate **violates** the query:



satisfies Q

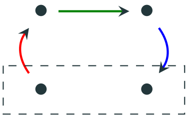
but

violates Q

$\rightarrow$ Reduce from $\mathrm{PQE}(Q_0)$ as we explained

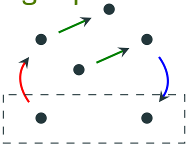
Rescuing the proof

We know that we have a **tight pattern**:



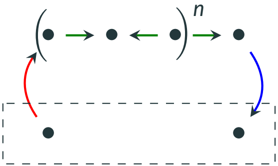
satisfies Q

but

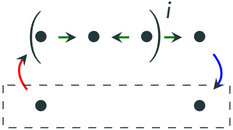


violates Q

Consider its **iterates** for each $n \in \mathbb{N}$:

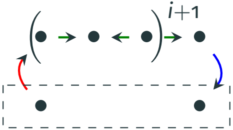


Case 1: some iterate **violates** the query:



satisfies Q

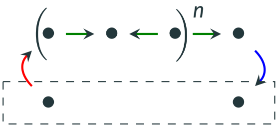
but



violates Q

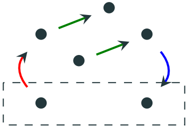
→ Reduce from $\mathrm{PQE}(Q_0)$ as we explained

Case 2: all iterates **satisfy** the query:



satisfies Q for all $n \in \mathbb{N}$

but



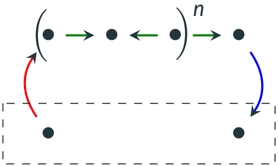
violates Q

Rescuing the proof

We know that we have a **tight pattern**:

satisfies Q but violates Q

Consider its **iterates** for each $n \in \mathbb{N}$:



Case 1: some iterate **violates** the query:

satisfies Q but violates Q

→ Reduce from $\mathrm{PQE}(Q_0)$ as we explained

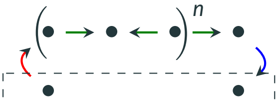
Case 2: all iterates **satisfy** the query:

satisfies Q for all $n \in \mathbb{N}$ but violates Q

→ Call this an **iterable pattern**

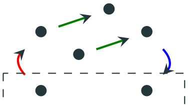
Using iterable patterns to show hardness of PQE

We have an **iterable pattern**:



satisfies Q for all $n \in \mathbb{N}$

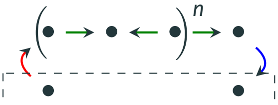
but



violates Q

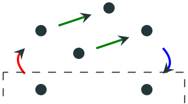
Using iterable patterns to show hardness of PQE

We have an **iterable pattern**:



satisfies Q for all $n \in \mathbb{N}$

but



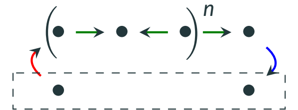
violates Q

Idea: reduce from the **#P-hard** problem **source-to-target connectivity**:

- Input: **undirected graph** with a **source** s and **target** t, all edges have probability $1/2$
- Output: what is the **probability** that the source and target are **connected**?

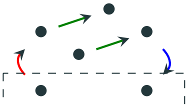
Using iterable patterns to show hardness of PQE

We have an **iterable pattern**:



satisfies Q for all $n \in \mathbb{N}$

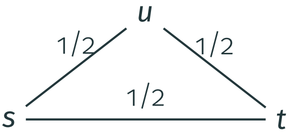
but



violates Q

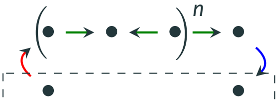
Idea: reduce from the **#P-hard** problem **source-to-target connectivity**:

- Input: **undirected graph** with a **source** s and **target** t, all edges have probability $1/2$
- Output: what is the **probability** that the source and target are **connected**?



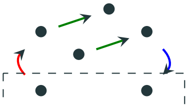
Using iterable patterns to show hardness of PQE

We have an **iterable pattern**:



satisfies Q for all $n \in \mathbb{N}$

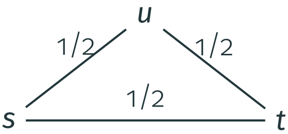
but



violates Q

Idea: reduce from the **#P-hard** problem **source-to-target connectivity**:

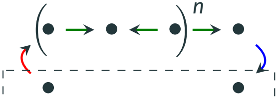
- Input: **undirected graph** with a **source** s and **target** t, all edges have probability $1/2$
- Output: what is the **probability** that the source and target are **connected**?



is coded as

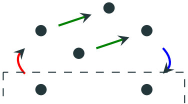
Using iterable patterns to show hardness of PQE

We have an **iterable pattern**:



satisfies Q for all $n \in \mathbb{N}$

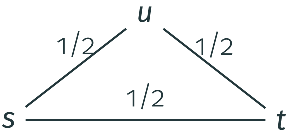
but



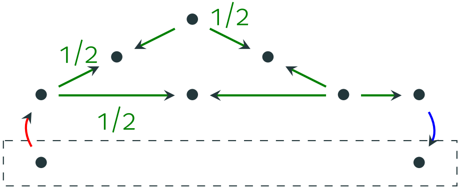
violates Q

Idea: reduce from the **#P-hard** problem **source-to-target connectivity**:

- Input: **undirected graph** with a **source** s and **target** t, all edges have probability $1/2$
- Output: what is the **probability** that the source and target are **connected**?

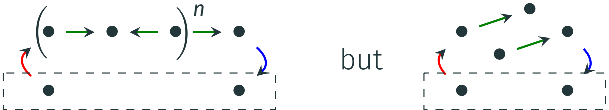


is coded as



Using iterable patterns to show hardness of PQE

We have an **iterable pattern**:



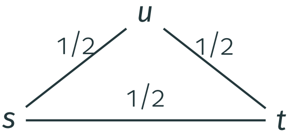
satisfies Q for all $n \in \mathbb{N}$

but

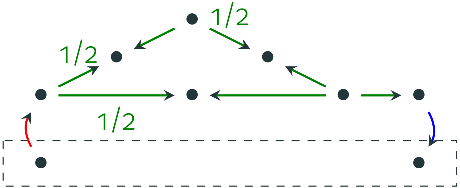
violates Q

Idea: reduce from the **#P-hard** problem **source-to-target connectivity**:

- Input: **undirected graph** with a **source** s and **target** t, all edges have probability $1/2$
- Output: what is the **probability** that the source and target are **connected**?



is coded as



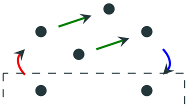
Idea: There is a **path connecting s and t** in a possible world of the graph at the left iff the query Q is **satisfied** in the corresponding possible world of the TID at the right

Using iterable patterns to show hardness of PQE

We have an **iterable pattern**:

satisfies Q for all $n \in \mathbb{N}$

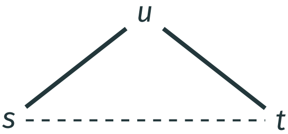
but



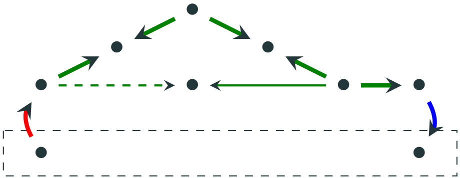
violates Q

Idea: reduce from the **#P-hard** problem **source-to-target connectivity**:

- Input: **undirected graph** with a **source** s and **target** t, all edges have probability $1/2$
- Output: what is the **probability** that the source and target are **connected**?



is coded as



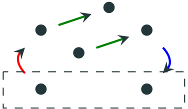
Idea: There is a **path connecting s and t** in a possible world of the graph at the left iff the query Q is **satisfied** in the corresponding possible world of the TID at the right

Using iterable patterns to show hardness of PQE

We have an **iterable pattern**:

satisfies Q for all $n \in \mathbb{N}$

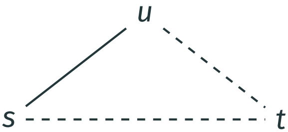
but



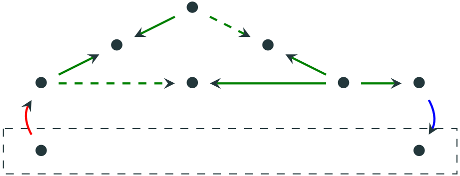
violates Q

Idea: reduce from the **#P-hard** problem **source-to-target connectivity**:

- Input: **undirected graph** with a **source** s and **target** t, all edges have probability $1/2$
- Output: what is the **probability** that the source and target are **connected**?



is coded as

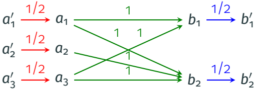


Idea: There is a **path connecting s and t** in a possible world of the graph at the left iff the query Q is **satisfied** in the corresponding possible world of the TID at the right

Proof technique

Hard part: show hardness for (variants of) the query Q: $x \longrightarrow y \longrightarrow z \longrightarrow w$

We reduce from $\mathrm{PQE}(Q)$, on **probabilistic graphs** G of the following form:

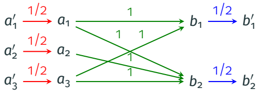


Task: count the number X of **red-blue edge subsets** that **violate** Q

Proof technique

Hard part: show hardness for (variants of) the query Q: $x \longrightarrow y \longrightarrow z \longrightarrow w$

We reduce from $\mathrm{PQE}(Q)$, on **probabilistic graphs** G of the following form:



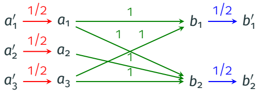
Task: count the number X of **red-blue edge subsets** that **violate** Q

- Split the **subsets** on some **parameter** e.g., the number of nodes: $X = X_1 + \cdots + X_k$

Proof technique

Hard part: show hardness for (variants of) the query Q: $x \longrightarrow y \longrightarrow z \longrightarrow w$

We reduce from $\mathrm{PQE}(Q)$, on **probabilistic graphs** G of the following form:



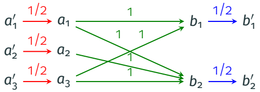
Task: count the number X of **red-blue edge subsets** that **violate** Q

- Split the **subsets** on some **parameter** e.g., the number of nodes: $X = X_1 + \cdots + X_k$
- Create unweighted copies of G modified with some **parameterized gadgets**
  - $\rightarrow$ Call the **oracle** for $\mathrm{SC(Q)}$ on each to get answers $N_1, \ldots, N_k$

Proof technique

Hard part: show hardness for (variants of) the query Q: $x \longrightarrow y \longrightarrow z \longrightarrow w$

We reduce from $\mathrm{PQE}(Q)$, on **probabilistic graphs** G of the following form:



Task: count the number X of **red-blue edge subsets** that **violate** Q

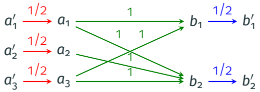
- Split the **subsets** on some **parameter** e.g., the number of nodes: $X = X_1 + \cdots + X_k$
- Create unweighted copies of G modified with some **parameterized gadgets**
  - → Call the **oracle** for $\mathrm{SC}(\mathrm{Q})$ on each to get answers $N_1, \ldots, N_k$
- Show that each N_i is a **linear function** of $X_1, \ldots, X_k$, so:

$$\begin{pmatrix} N_1 \\ \vdots \\ N_k \end{pmatrix} = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,k} \\ \vdots & \ddots & \vdots \\ \alpha_{k,1} & \cdots & \alpha_{k,k} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ \vdots \\ X_k \end{pmatrix}$$

Proof technique

Hard part: show hardness for (variants of) the query Q: $x \longrightarrow y \longrightarrow z \longrightarrow w$

We reduce from $\mathrm{PQE}(Q)$, on **probabilistic graphs G** of the following form:



Task: count the number X of **red-blue edge subsets** that **violate** Q

- Split the **subsets** on some **parameter** e.g., the number of nodes: $X = X_1 + \cdots + X_k$
- Create unweighted copies of G modified with some **parameterized gadgets**
  - $\rightarrow$ Call the **oracle** for $\mathrm{SC}(\mathrm{Q})$ on each to get answers $N_1, \ldots, N_k$
- Show that each N_i is a **linear function** of $X_1, \ldots, X_k$, so:

$$\begin{pmatrix} N_1 \\ \vdots \\ N_k \end{pmatrix} = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,k} \\ \vdots & \ddots & \vdots \\ \alpha_{k,1} & \cdots & \alpha_{k,k} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ \vdots \\ X_k \end{pmatrix}$$

- Show **invertibility** of this matrix to recover the X_i from the N_i

Using the equation system

We have obtained the system:

$$\begin{pmatrix} N_1 \\ \vdots \\ N_k \end{pmatrix} = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,k} \\ \vdots & \ddots & \vdots \\ \alpha_{k,1} & \cdots & \alpha_{k,k} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ \vdots \\ X_k \end{pmatrix}$$

Using the equation system

We have obtained the system:

$$\begin{pmatrix} N_1 \\ \vdots \\ N_k \end{pmatrix} = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,k} \\ \vdots & \ddots & \vdots \\ \alpha_{k,1} & \cdots & \alpha_{k,k} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ \vdots \\ X_k \end{pmatrix}$$

- The oracle for MC has given us $N_1, \ldots, N_k$

Using the equation system

We have obtained the system:

$$\begin{pmatrix} N_1 \\ \vdots \\ N_k \end{pmatrix} = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,k} \\ \vdots & \ddots & \vdots \\ \alpha_{k,1} & \cdots & \alpha_{k,k} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ \vdots \\ X_k \end{pmatrix}$$

- The oracle for MC has given us $N_1, \ldots, N_k$
- We need $X = X_1 + \cdots + X_k$ to solve PQE and finish the reduction

Using the equation system

We have obtained the system:

$$\begin{pmatrix} N_1 \\ \vdots \\ N_k \end{pmatrix} = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,k} \\ \vdots & \ddots & \vdots \\ \alpha_{k,1} & \cdots & \alpha_{k,k} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ \vdots \\ X_k \end{pmatrix}$$

- The **oracle** for **MC** has given us $N_1, \ldots, N_k$
- We **need** $X = X_1 + \cdots + X_k$ to solve **PQE** and finish the reduction

→ If the matrix is **invertible**, then we have succeeded

Using the equation system

We have obtained the system:

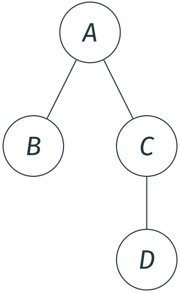
$$\begin{pmatrix} N_1 \\ \vdots \\ N_k \end{pmatrix} = \begin{pmatrix} \alpha_{1,1} & \cdots & \alpha_{1,k} \\ \vdots & \ddots & \vdots \\ \alpha_{k,1} & \cdots & \alpha_{k,k} \end{pmatrix} \cdot \begin{pmatrix} X_1 \\ \vdots \\ X_k \end{pmatrix}$$

- The **oracle** for **MC** has given us $N_1, \ldots, N_k$
- We **need** $X = X_1 + \cdots + X_k$ to solve **PQE** and finish the reduction

→ If the matrix is **invertible**, then we have succeeded

We can choose gadgets and parameters to get a **Vandermonde matrix**, and show invertibility via several **arithmetical tricks**

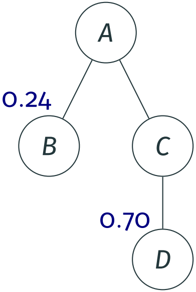
The semistructured model and XML



```
<a>
  <b>...</b>
  <c>
    <d>...</d>
  </c>
</a>
```

- Tree-like structuring of data
- No (or less) schema constraints
- Allow mixing tags (structured data) and text (unstructured content)
- Particularly adapted to tagged or heterogeneous content

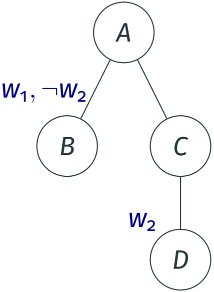
Simple probabilistic annotations



- **Probabilities** associated to tree nodes
- Express parent/child dependencies
- Impossible to express more complex dependencies
- ⇒ some **sets of possible worlds** are not expressible this way!

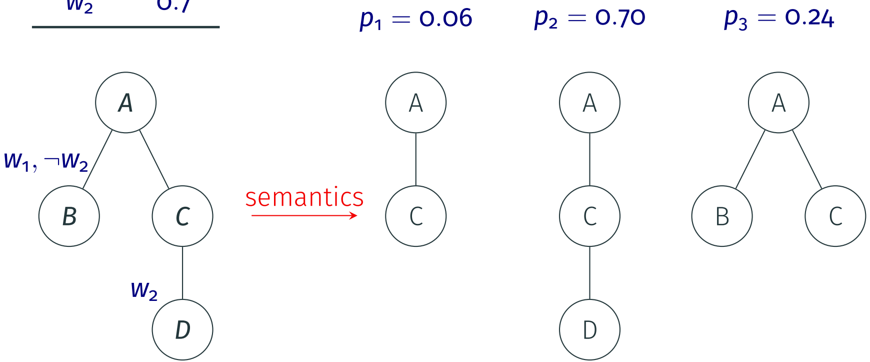
Annotations with event variables

Event	Prob.
w_1	0.8
w_2	0.7



Annotations with event variables

Event	Prob.
w_1	0.8
w_2	0.7



- Expresses **arbitrarily complex** dependencies

Query evaluation on probabilistic XML

- Query evaluation for probabilistic XML: what is the probability that a (fixed) **tree automaton** accepts?

Query evaluation on probabilistic XML

- Query evaluation for probabilistic XML: what is the probability that a (fixed) **tree automaton** accepts?
- Can be computed **bottom-up** in the simple model [Cohen et al., 2009]

Query evaluation on probabilistic XML

- Query evaluation for probabilistic XML: what is the probability that a (fixed) **tree automaton** accepts?
- Can be computed **bottom-up** in the simple model [Cohen et al., 2009]
- **#P-hard** in the general model

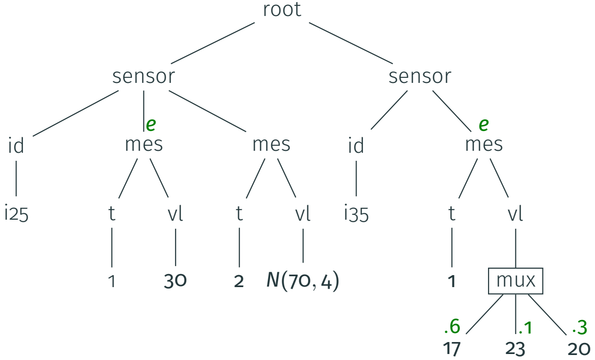
Query evaluation on probabilistic XML

- Query evaluation for probabilistic XML: what is the probability that a (fixed) **tree automaton** accepts?
- Can be computed **bottom-up** in the simple model [Cohen et al., 2009]
- **#P-hard** in the general model
- This generalizes to PQE for **MSO** on **relational databases (TID)** when assuming that the **treewidth** is bounded [Amarilli et al., 2015]

Query evaluation on probabilistic XML

- Query evaluation for probabilistic XML: what is the probability that a (fixed) **tree automaton** accepts?
- Can be computed **bottom-up** in the simple model [Cohen et al., 2009]
- **#P-hard** in the general model
- This generalizes to PQE for **MSO** on **relational databases (TID)** when assuming that the **treewidth** is bounded [Amarilli et al., 2015]
- Bounding the treewidth is **necessary** for tractability in a certain sense [Amarilli et al., 2016]

A general probabilistic XML model [Abiteboul et al., 2009]

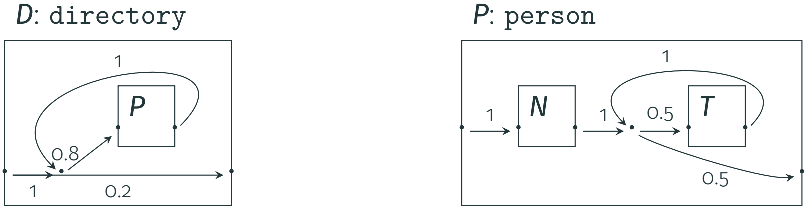


- *e*: event "it did not rain" at time 1
- mux: mutually exclusive options
- $N(70, 4)$: normal distribution

- Compact representation of a **set of possible worlds**
- Two kinds of dependencies: global (*e*) and local (mux)
- Generalizes **all previously proposed models** of the literature

Recursive Markov chains [Benedikt et al., 2010]

```
<!ELEMENT directory (person*)>
<!ELEMENT person (name,phone*)>
```



- Probabilistic model that **extends** PXML with local dependencies
- Generate documents of **unbounded** width or depth

C-tables [Imielinski and Lipski, 1984]

Patient	Examin. 1	Examin. 2	Diagnosis	Condition
A	23	12	α	
B	10	23	$\bot_1$	
C	2	4	γ	
D	$\bot_2$	15	$\bot_1$	
E	$\bot_3$	17	β	$18 < \bot_3 < \bot_2$

- NULLs are labeled, and can be **reused** inside and across tuples
- **Arbitrary correlations** across tuples
- **Closed** under the relational algebra
- Every set of possible worlds can be represented as a database with c-tables